AD-A015 141

A COMPUTER PROGRAM FOR THE PREDICTION OF SOLID PROPELLANT ROCKET MOTOR PERFORMANCE. VOLUME II

D. E. Coats, et al

Ultrasystems, Incorporated

Prepared for:

Air Force Rocket Propulsion Laboratory

July 1975

AFRPL-TR-75-36

A COMPUTER PROGRAM FOR THE PREDICTION OF SOLID PROPELLANT ROCKET MOTOR PERFORMANCE, VOL. II

FINAL REPORT

Ultrasystems, Inc.
2400 Michelson Drive
Irvine, California 92664

Authors: D. E. Coats
J. N. Levine
G. R. Nickerson
T. J. Tyson
N. S. Cohen
D. P. Harry III
C. F. Price

JULY 1975

APPROVED FOR PUBLIC RELEASE;

DISTRIBUTION UNLIMITED

AIR FORCE ROCKET PROPULSION LABORATORY
DIRECTOR OF SCIENCE AND TECHNOLOGY
AIR FORCE SYSTEMS COMMAND
EDWARDS, CALIFORNIA 93523

NOTICES

"When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications or other data, is not to be regarded by implication or otherwise, or in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto."

FOREWORD

This report was submitted by Ultrasystems, Inc., Environmental and Applied Sciences Division, 2400 Michelson Drive, Irvine, California 92664, under Contract No. F04611-73-C-0038, Job Order No. 305909LZ with the Air Force Rocket Propulsion Laboratory, Edwards, CA 93523.

This report consists of three volumes. Volume I describes a computer program for the prediction of Solid Propellant Rocket Motor Performance. The computer program described herein will be referred to as the SPP program, and describes the engineering analysis which was used in developing this computer program and the results obtained to date.

Volume II of this report is a programming document of the computer program which was developed under this contract. It includes a subroutine-by-subroutine description of all of the elements of the SPP program.

Volume III of this report is a Program User's Manual which describes the input necessary to execute the SPP computer program and the information required to interpret the output. A sample case is also included.

This report has been reviewed by the Information Office/DOZ and is releasable to the National Technical Information Service (NTIS). At NTIS it will be available to the general public, including foreign nations.

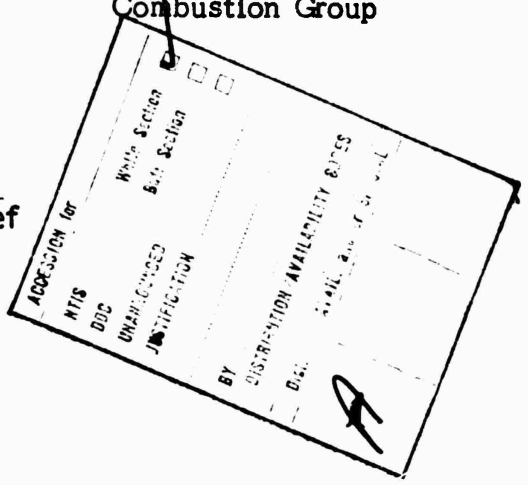
This report is unclassified and suitable for public release.

JOHN L. WILLIAMS, Lt., USAF
Project Engineer

W. C. ANDREPONT, GS-14, Chief
Combustion Group

FOR THE COMMANDER

W.S. ANDERSON, GS-15, Acting Chief
Technology Division



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFRPL-TR-75-36	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Computer Program for the Prediction of Solid Propellant Rocket Motor Performance, Vol. I, II, and III		5. TYPE OF REPORT & PERIOD COVERED Final Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) D. E. Coats, N. S. Cohen J. N. Levine, D. P. Harry III, et al (Ultrasystems, Inc.) (Lockheed Propulsion)		8. CONTRACT OR GRANT NUMBER(s) F04611-73-C-0038
9. PERFORMING ORGANIZATION NAME AND ADDRESS Ultrasystems, Inc. 2400 Michelson Drive Irvine, California 92664		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS JON 305909LZ
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Rocket Propulsion Laboratory/DY Edwards, CA 93523		12. REPORT DATE July 1975 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Performance prediction
Solid Rocket Motors
Ballistic Analysis
Specific impulse losses

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A flexible, modular, fully automated, solid rocket motor performance prediction program has been developed. The program, which has been given the acronym SPP is based on six pre-existing computer codes. These codes have been integrated and modified, as required. To supplement the theory, where necessary, and to increase the flexibility of the program, a number of existing and newly developed semi-empirical correlations were incorporated into the program. The program has a general three-dimensional grain design capability, coupled to a one-dimensional ballistics analysis. The deviations from ideal performance are

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

computed as a series of independent efficiencies. The program currently treats the following losses: two-dimensional/two-phase (coupled), nozzle erosion, kinetics, boundary layer, combustion efficiency, submergence. The program predicts average delivered performance, as well as mass flow, pressure, thrust, impulse, and specific impulse as functions of time and trajectory.

In order to assess the validity of the SPP program, calculated results were compared to firing data for four different types of motors. While conclusive statements regarding the accuracy and range of validity of the SPP program cannot be made until additional verification efforts are conducted, the results of these four test cases were encouraging. These calculations also served to demonstrate the desirability of eliminating some of the present limitations of the program.

PREFACE

This is Volume II of a three-part report which describes a computer program for the prediction of Solid Propellant Rocket Motor Performance. The computer program described herein will be refered to as the SPP program.

Volume I of this report describes the engineering analysis which was used in developing this computer program.

Volume II of this report is a programming document of the computer program which was developed under this contract. It includes a subroutine-by subroutine description of all of the elements of the SPP program.

Volume III of this report is a Program User's Manual which describes the input necessary to execute the SPP computer program and the information required to interpret the output. A sample case is also included.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

NOMENCLATURE

A	-	Nozzle area
A_b	-	Propellant burn area
A_p	-	Port area
C^*	-	Characteristic exhaust velocity
C_D	-	Nozzle discharge coefficient
C_H	-	Film coefficient
c_i	-	Mass fraction of i^{th} species
C_p	-	Specific heat of gas
C_{p_ℓ}	-	Specific heat of liquid condensed phase
C_{p_s}	-	Specific heat of solid condensed phase
C_s	-	Specific heat of propellant
D	-	Diameter
D_p	-	Particle diameter
F	-	Thrust
g	-	Gravitational constant
h	-	Enthalpy
ΔH_m	-	Heat of fusion
I	-	Impulse
I_{sp}	-	Specific impulse
K	-	Thermal conductivity
L^*	-	Characteristic length of motor
m	-	Mass
$\dot{m}$	-	Nozzle mass flow rate
M	-	Molecular weight, also Mach No.
n	-	Burning rate pressure exponent
P	-	Pressure
$\dot{P}$	-	Rate of change of pressure
Pr	-	Prandtl number
r	-	Propellant burning rate, also radius
r^*	-	Throat radius
R	-	Gas constant
$\mathcal{R}$	-	Universal gas constant
$\dot{r}_t$	-	Nozzle erosion rate

t	-	Time
T	-	Temperature
u	-	Gas velocity
v	-	Gas velocity
w	-	Propellant web
$\dot{w}$	-	Weight flow
X_i	-	Mole fraction
x	-	Axial distance

Subscripts

a	-	Ambient
BL	-	Boundary Layer
c	-	Chamber
CE	-	Combustion efficiency
D	-	Delivered
D_a	-	Delivered to ambient
DIV	-	Divergence
e	-	Nozzle exit, or boundary layer edge
F	-	Flame temperature
i	-	Initial, t=0
I	-	Insulation
KIN	-	Kinetics
ℓ	-	Liquid phase
m	-	Melting points of condensed phase
ODK	-	Based on ODK program results
P	-	Refers to all particle groups, liquid and solid
RE	-	Restricted equilibrium
s	-	Slot, or solid phase
SUB	-	Submergence
T	-	Total
t	-	Throat
TBL	-	Based on TBL program results
th	-	Theoretical

Greek Symbols

α	-	Thermal diffusivity
α'_c	-	Nozzle erosion parameter
γ	-	Isentropic exponent
δ^*	-	Boundary layer displacement thickness
ϵ	-	Nozzle expansion ratio
η	-	Fractional loss efficiency
θ	-	Boundary layer momentum thickness, also angles
λ	-	Emissivity, also reaction order
μ	-	Viscosity, also Mach angle
ξ	-	Mole fraction of condensed phase
ρ	-	Density
σ	-	Stefan-Boltzman constant

Super Scripts

$\bar{}$	-	Average values

1.0 Introduction

This document, Volume II, provides a detailed description of all of the elements used in the Solid Propellant Rocket Motor Performance Prediction (SPP) computer program. The goal in developing the SPP code was to develop a computerized analysis technique that could predict solid propellant rocket motor delivered specific impulse to within ±2% and delivered thrust and total impulse to within ±5%.

The contents of this volume are not intended to supply the reader with a complete description of the SPP computer code. Instead, it is intended to give, along with the information supplied in Volumes I and III, the necessary information about the internal functions of the SPP code to allow the educated user to understand and/or modify the basic internal workings of this computer program.

The SPP code consists of five basic computational modules plus a master control module which selects via user input which of the computational modules are to be executed. The approach used in developing this code has been to supply the user with the maximum amount of flexibility consistent with ease of use.

The five basic computational modules used in the SPP code are described very briefly below in Table 1-1. A more detailed description of these modules is presented in Volume I and to a lesser extent in Volume III.

Module	Description
ØDE	Calculates the reference or idealized performance of the solid propellant rocket motor.
BAL	Calculates the grain shape and internal ballistics parameters as a function of time.
ØDK	Calculates the nozzle performance considering the degradation in I_{sp} due to finite rate chemical kinetics
TD2P	Calculates the nozzle performance considering the degradation in performance due to two dimensional and two phase flow effects.
TBL	Calculates the nozzle performance considering the degradation in performance due to a viscous turbulent boundary layer.

Table 1-1 Basic Computation Modules

Volume I consists of the engineering analysis incorporated in the SPP code while Volume III describes the input necessary to execute the SPP code. The user's manual (Volume III) and this volume are considered to be the prerequisite amount of information needed to successfully understand and/or modify the SPP program.

Section 2 of this volume describes the overall flow of the SPP program along with the data communicated between the various computational modules. The overlay structure and labeled common blocks used by each subprogram are detailed in section 3 while the linkages and files used by the SPP code are described in section 4. A complete subroutine by subroutine description of this code is included in section 5. Section 5 is orgainzed by overlay structure and includes a description of the main routine of each overlay followed by a description of each subprogram in that overlay in alphabetical order.

Liberal use of existing documentation, References 1 through 6, about the individual elements of the SPP code has been used in this volume. The interested reader would be well advised to obtain these documents if he does not already have access to them.

2.0 SPP Program Flow Charts

This section of Volume II, the Computer Program Description, is intended to give the user of the SPP code a general description of the overall flow of the computer program. Also described here is how data is transmitted between modules when the many options which are available in the SPP code are executed.

Figure 2-1 shows the overall logical flow of the SPP code. The solid lines in this figure show the mandatory flow while the dashed lines indicate the optional paths the computer program can take. In the case that more than one optional path can be taken, a diamond sign, ◇ , is used to indicate the alternate paths.

The boxes on the right hand side of Figure 2-1 show the action taken by the SPP code in response to input directive cards and also to the selection of loss modules to be executed. The items on the left hand side of this figure show the alternate logical execution paths which are determined by user input.

Figure 2-2 shows the actual execution sequence taken by the SPP code. The following notes are intended to clarify Figure 2-2.

Alternate linkage data read in under the control of subroutine PRØB is read in in sequence (See Volume III, Section 2.11).

Linkage data written by the various loss modules is positioned by subroutine SETUP before being read by subroutine PICKUP. This insures that none of the linkage data is overwritten unless multiple cases are executed (in which case, the linkage data for the first case is destroyed).

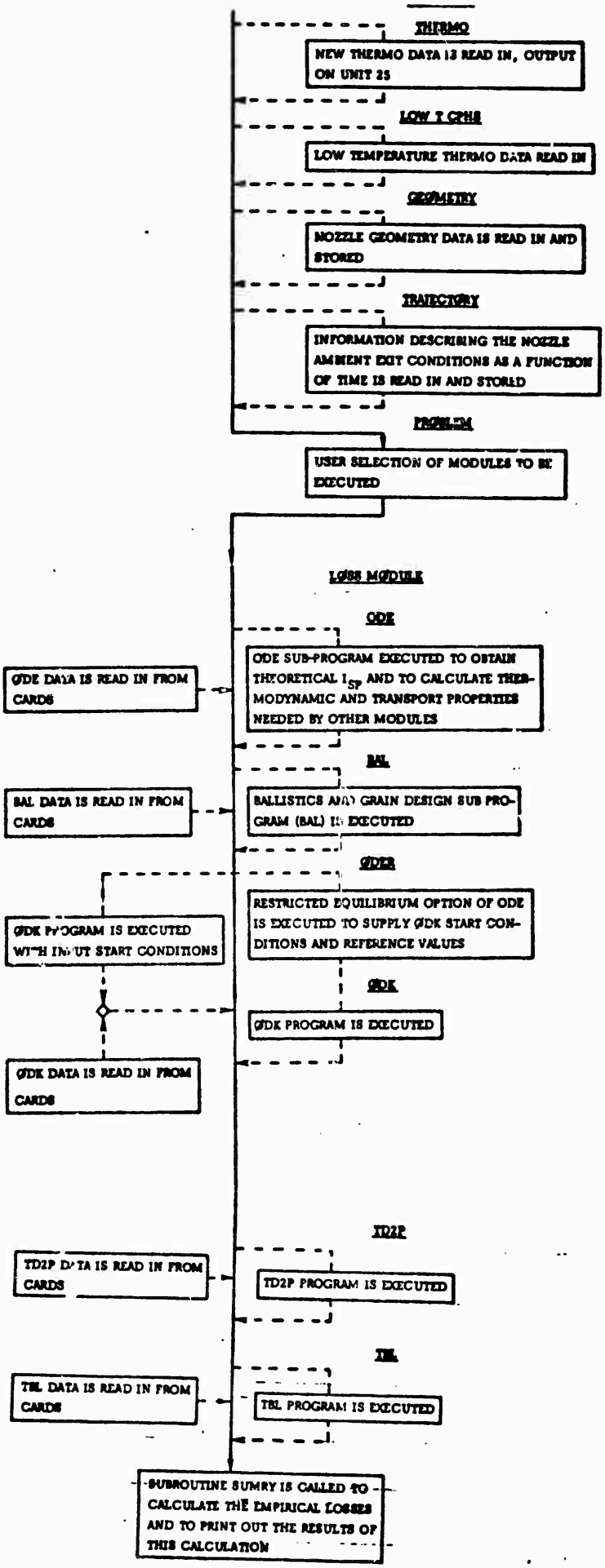
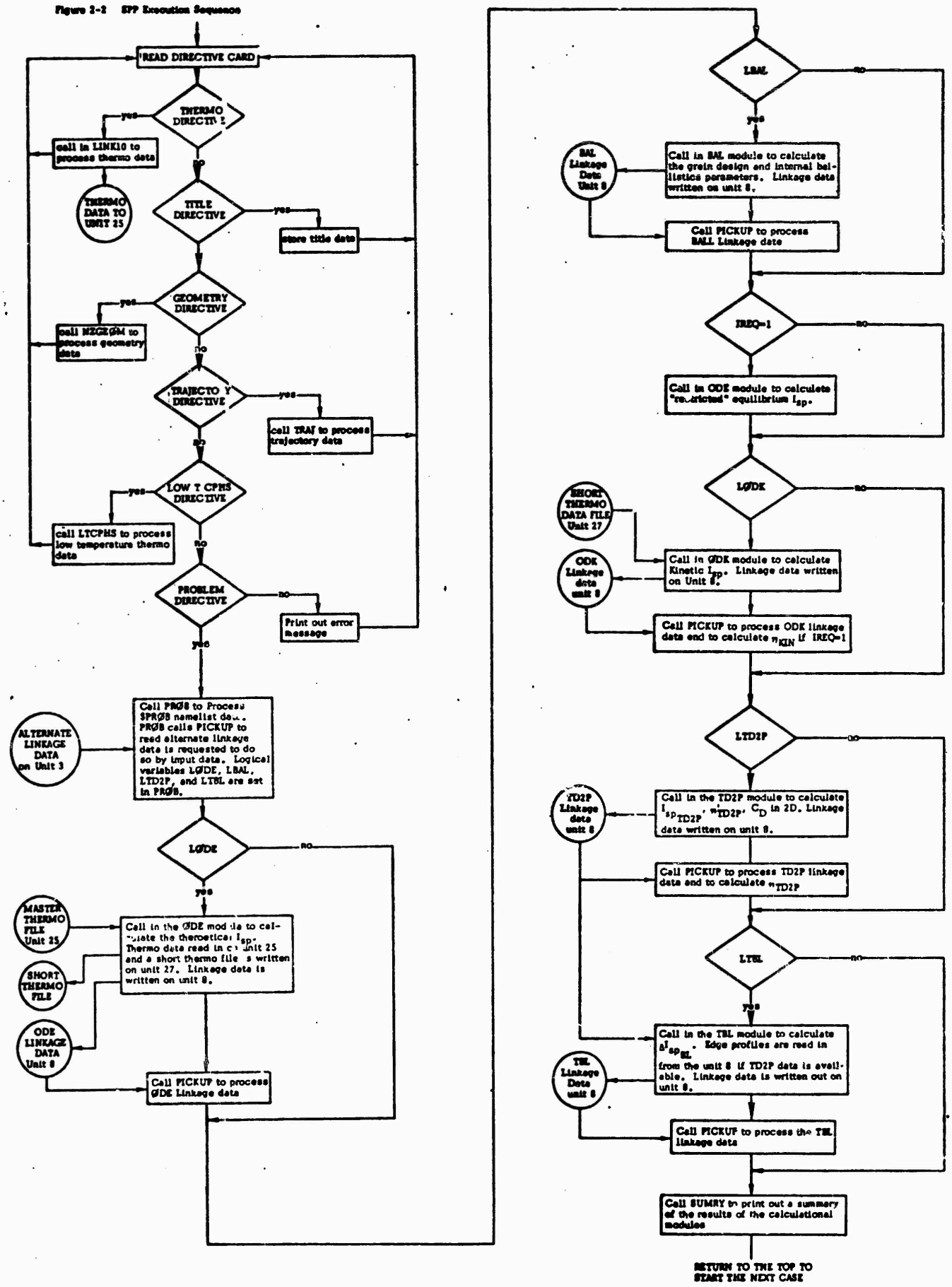


Figure 2-1 IPP Overall Flow Diagram

Figure 2-2 SPP Execution Sequence



3.0 Overlay and Common Block Description

Section 3.1 describes the overlay structure of the SPP while Section 3.2 describes the common blocks used in the SPP code.

3.1 Program Overlay Structure

The SPP Program overlay structure is shown in Table 3-1 and corresponds to the order of appearance of the subroutine write ups in Section 5, if read downward and then to the right.

This overlay structure was developed under the restrictions of the CDC 6000 series loader (SCOPE 3.3) which allows only three levels of overlay. Considerable savings in central memory storage can be achieved with a more extensive overlay structure is used.

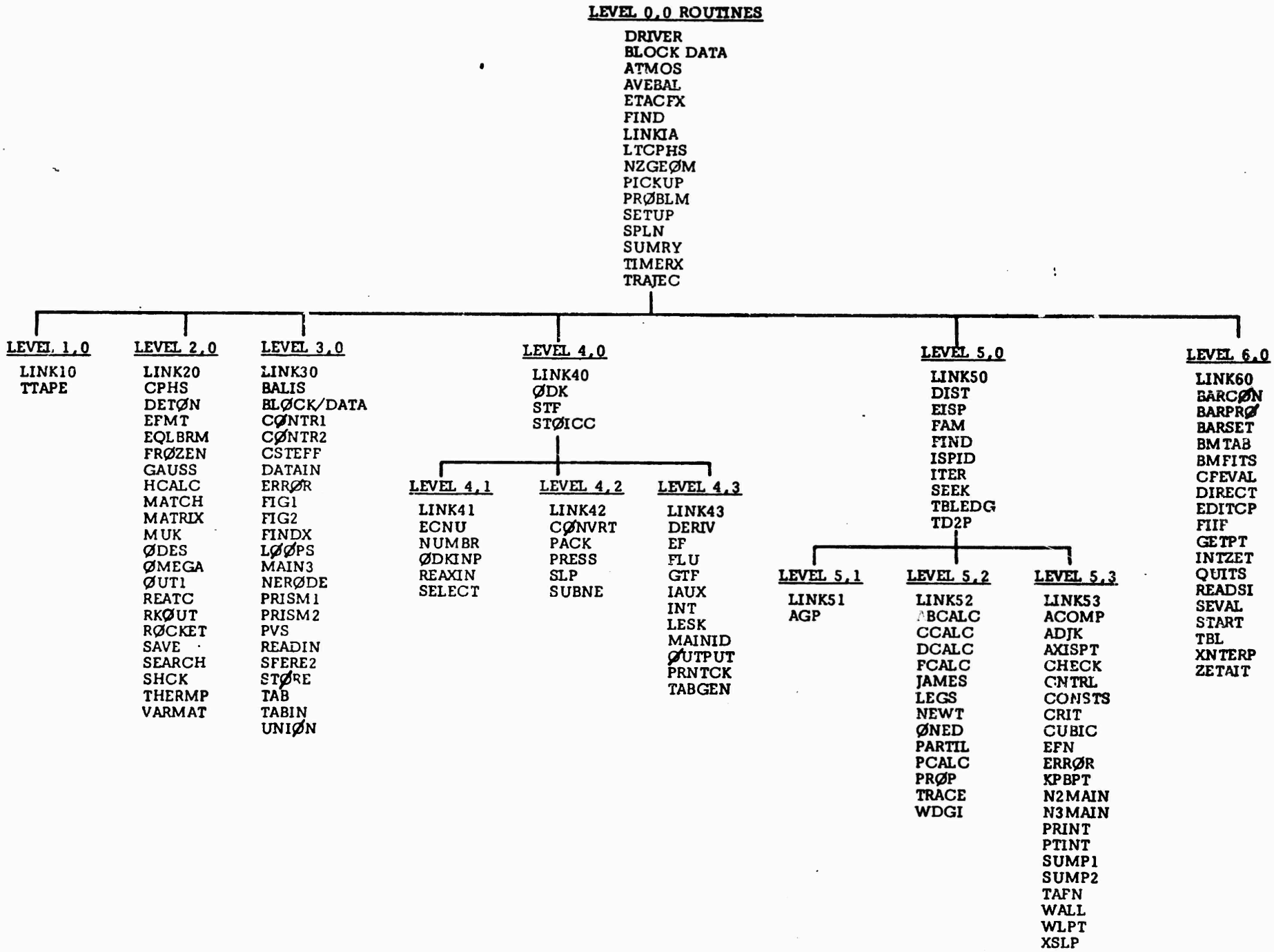


Table 3-1 Program Overlay Structure

3.2 Common Blocks

The common blocks and the subroutines that are referred to by, are listed in Table 3-2. Tables 3-3 through 3-14 list the variables in some of the more important labeled common blocks used in the SPP code. These tables are organized alphabetically by common block name.

Appendix B of Reference 1 contains a dictionary of the variables appearing in labeled common blocks in the ØDE module. While there are some minor variances between Reference 1 and the SPP code, this dictionary should be very useful to anyone attempting to modify the ODE module. Unfortunately, similiar dictionaries are not available for the other modules.

Table 3-2 Common Block References

COMMON BLOCK	REFERRED TO BY SUBROUTINE
ADD	BLKDAT CONTR2 FIG1 FIG2 LOOPS PRISM2 UNION
AMACHR	FAM TBLEDG
ANMEL	PRINT
ARODE	BLKDAT DRIVER ODES ODKINP OUTPUT OUT1 PACK REACT RKTOUT SEARCH
ARPRNT	NZGEOM ODES ODKINP PICKUP

COMMON BLOCK	REFERRED TO BY SUBROUTINE
	SUMRY
BALCOM	AVEBAL NZGEOM PICKUP SUMRY
BALL	BALIS BLKDAT LOOPS MAIN3 PVS READIN
CARD	READIN STORE
CDELHX	DRIVER ODES SAVE
CFIIF	START ZETAIT
CIDATA	BARCON BARPRO BARSET GETPT QUITS READSI SEVAL START
CINT	IAUX INT LESK MAIN1D
CK	ERROR READIN
CODATA	BARCON BARPRO BARSET EDITCP GETPT QUITS READSI START ZETAIT

COMMON BLOCK	REFERRED TO BY SUBROUTINE
CODE	BLKDAT ERROR READIN STORE
CODTDK	FLU MAIN1D ODK ODKINP
COEFSX	CPHS SEARCH
COFIIF	BARPRO FIIF QUITS
COMCAS	CONVRT DERIV DRIVER EF FLU GTF IAUX INT MAIN1D ODKINP OUTPUT PACK PRES READIN SELECT STF
COMERR	TTAPE
COMXP	DRIVER ODES
COMY	CONVRT DERIV DRIVER EF FLU GTF IAUX INT MAIN1D

COMMON BLOCK	REFERRED TO BY SUBROUTINE
	ODKINP OUTPUT PACK PRE3 REAXIN STF
COM1	BLKDAT TAB TABIN
COM2	BLKDAT MAIN3 READIN
COM3	BLKDAT MAIN3 READIN
COM4	BLKDAT TAB TABIN
COM5	BLKDAT DATAIN READIN TAB TABIN
CPEER	CPHS ROCKET
CPRNT	DRIVER INT MAIN1D ODKINP PACK
CSEVAL	BARPRO BARSET GETPT QUITS READSI SEVAL START
CSPRXC	CONVRT DERIV EF PACK REAXIN

COMMON BLOCK	REFERRED TO BY SUBROUTINE
	SELLCT
	STOICC
CUTIL	CONVRT
	CPHS
	DRIVER
	FLU
	IAUX
	MAIN1D
	MUK
	ODES
	ODK
	ODKINP
	OUTPUT
	OUT1
	PACK
	PICKUP
	PRES
	REACT
	REAXIN
	RKTOUT
	ROCKET
	SEARCH
	SELECT
	SPLN
	STF
	SUMRY
	TTAPE
CWALL	DRIVER
	FLU
	MAIN1D
	ODKINP
	PACK
	PRES
DOLOOP	DERIV
	EF
	IAUX
	INT
	MAIN1D
	ODKINP †
	PRNTCK
DOUBLE	EQLBRM
	GAUSS
	MATRIX
	ODES
	OUT1
EDGEX	BARPRO

COMMON BLOCK	REFERRED TO BY SUBROUTINE
	READ?I
EDGEY	BARPRO GETPT READ?I
EEE	PROBLM SUMRY
FRR	ADJK AGP CNTRL ERROR FCALC JAMES KPBPT N2MAIN PARTIL TD2P WALL WLPT
EXTRAP	DRIVER FIND PICKUP SUMRY
FIG	BLKDAT CONTR1 CONTR2 FIG1 FIG2 PRISM1 PRISM2 READIN STORE
FNALBT	DERIV FLU IAUX MAIN1D
HHTCOM	EQLBRM ODES
IDOL	AGP EISP ISP1D

COMMON BLOCK	REFERRED TO BY SUBROUTINE
INDX	CPHS DRIVER EQLBRM FROZEN GAUSS HCALC MATRIX ODES OUT1 REACT RKTOUT ROCKET SAVE SEARCH
INDXX	CPHS EQLBRM FROZEN GAUSS HCALC MATRIX ODES REACT RKTOUT ROCKET SAVE
INERTS	REAXIN SELECT
IUSE	CPHS DRIVER EQLBRM FROZEN HCALC MATRIX ODES OUT1 RKTOUT ROCKET SAVE SEARCH SELECT
KINFU	DRIVER FROZEN MAIN1D ODES ODKINP OUTPUT

COMMON BLOCK	REFERRED TO BY SUBROUTINE
	OUT1
	PACK
	PRES
	REACT
	REAXIN
	RKTOUT
	ROCKET
	SELECT
LKOM	BALIS
	MAIN3
	NZGEOM
	PICKUP
	PROBLM
	SUMRY
LKEQKN	DRIVER
	ODES
	ODK
	PICKUP
	PROBLM
	RKTOUT
	ROCKET
	SUMRY
LKGEOM	NZGEOM
	ODKINP
	SUMRY
	TD2P
LKMELT	DERIV
	MAIN1D
LKOJK	PICKUP
	SUMRY
LKTBL	PROBLM
	READSI
LKTD2P	AGP
	NZGEOM
	PICKUP
	PROBLM
	READSI
	TD2P
LOOP	BALIS
	BLKDAT
	CONTR2
	ERROR
	FIG1

COMMON BLOCK	REFERRED TO BY SUBROUTINE
	FIG2
	LOOP3
	PRISM1
	PRISM2
	READIN
	STORE
	UNION
LOWTN	CPHS
	DRIVER
	LTCPHS
	STF
MACHDR	DERIV
	FLU
MAN	PROBLM
	SUMRY
MARKET	CONVRT
	DRIVER
	FLU
	MAIN1D
	NZGEOM
	ODES
	ODK
	ODKINP
	OUTPUT
	PACK
	PICKUP
	PROBLM
	REAXIN
	ROCKET
	SELECT
	SUMRY
MISC	BLKDAT
	CPHS
	DRIVER
	EQLBRM
	FROZEN
	HCALC
	MATRIX
	ODES
	OUT1
	PACK
	REACT
	RKTOUT

COMMON BLOCK	REFERRED TO BY SUBROUTINE
	ROCKET SAVE SEARCH
MISCX	CPHS EQLBRM FROZEN HCALC MATRIX ODES REACT RKTOUT ROCKET SAVE
MOLWTS	OUT1 ROCKET SEARCH
MUST	DERIV FLU
NAMD	AGP EISP I8P1D TD2P
NAME2	ABCALC CCALC FCALC JAMES PARTIL PCALC PROP TD2P TRACE

COMMON BLOCK	REFERRED TO BY SUBROUTINE
NAMED	ACOMP AGP AXISPT CNTRL CONSTS EFN EISP ISP1D KPBPT ONED PARTIL PRINT PTINT SUMP1 SUMP2 TBLEDG TD2P TRACE WALL WDGI WLPT
NAMEF	ACOMP ONED PARTIL TD2P TRACE
NAMEI	ADJK CNTRL KPBPT PRINT TD2P
NAMEL	AGP ONED PARTIL PROP TD2P TRACE WDGI

COMMON BLOCK	REFERRED TO BY SUBROUTINE
NAMEM	ACOMP AXISPT CHECK CNTRL CONSTS KPBPT ONED PARTIL PRINT PTINT SUMP1 SUMP2 TD2P TRACE WALL WDGI WLPT
NAMEN	ACOMP AXISPT EFN KPBPT PTINT SUMP1 SUMP2 TD2P
NAMEP	ACOMP ADJK AXISPT CHECK CNTRL CONSTS EFN KPBPT PRINT PTINT SUMP1 SUMP2 TD2P WLPT

COMMON BLOCK	REFERRED TO BY SUBROUTINE
NAMEQ	ACOMP CNTRL CONSTS ONED PARTIL PRINT TD2P TRACE
NAMER	ONED PARTIL PROP TD2P TRACE
NAMES	ONED PARTIL PROP TD2P TRACE WDGI
NAMET	AXISPT KPBPT PTINT SUMP1 SUMP2 TD2P WLPT
NAMEW	ODKINP * TD2P WALL
NAMEX	ONED PARTIL PRINT PROP TBLEDG TD2P TRACE
NAMEY	ONED PARTIL PROP TD2P TRACE
NAMEZ	PARTIL TD2P TRACE

COMMON BLOCK	REFERRED TO BY SUBROUTINE
NAMF1	ABCALC CCALC FCALC JAMES ONED PARTIL PCALC TD2P
NAME2	ABCALC CCALC FCALC JAMES PCALC
NAME3	ABCALC CCALC FCALC JAMES PARTIL PCALC PROP TRACE
NAME4	ABCALC CCALC
NAMIN	MAIN1D ODKINP
NIN	BLKDAT FRROR READIN STORE
ODECUT	OUT1 RKTOUT ROCKET

COMMON BLOCK	REFERRED TO BY SUBROUTINE
ODKCOM	CONVRT DERIV DRIVER EF FLU IAUX INT MAINID ODES ODK ODKINP OUTPUT OUT1 PACK PRES PRNTCK REAXIN RKTOUT ROCKET SELECT
ODKHX	CONVRT DERIV EF ODK PACK
ODKSP	CONVRT DERIV DRIVER EF FLU GTF IAUX MAINID ODKINP OUTPUT PACK REAXIN STF
OUPT	BLKDAT FFMT FROZEN OUT1 RKTOUT VARFMT

COMMON BLOCK	REFERRED TO BY SUBROUTINE
PERF	DRIVER EQLBRM FROZEN ODES OUT1 RKTOUT ROCKET
PERFX	EQLBRM FROZEN ODES OUT1 RKTOUT ROCKET
PMFAN	BARSET
POINTS	DRIVER EQLBRM FROZEN HCALC MATRIX ODES OUT1 RKTOUT ROCKET
POINTX	EQLBRM FROZEN HCALC MATRIX ODES RKTOUT ROCKET
PTABLE	FLU IAUX MAIN1D ODK ODKINP PRES
PTSAVE	DRIVER ODES
RDCARD	DRIVER PROBLM

COMMON BLOCK	REFERRED TO BY SUBROUTINE
RDIREX	DRIVER ODES PACK REAXIN SEARCH SELECT STOICC
RVACOM	INT MAINID ODK ODKINP OUTPUT
SAVE	CONTR1 FIG2 PRISM1
SAVODE	DRIVER ODES
SHGAMA	DERIV ODK ODKINP
SPECES	CPHS EQLBRM FROZEN HCALC MATRIX ODES OUT1 RKTOUT ROCKET SAVE SEARCH
SPECEX	CPHS EQLBRM FROZEN HCALC MATRIX ODES OUT1 RKTOUT ROCKET SAVE SEARCH

COMMON BLOCK	REFERRED TO BY SUBROUTINE
SPINFO	CPHS DRIVER EQLBRM ODES ODKINP OUT1 PACK READSI RKTOUT ROCKET SEARCH SELECT SUMRY TD2P
TBLDT1	DRIVER IAUX MAIN1D ODKINP
TBLDT2	DRIVER IAUX MAIN1D
TBLRE	PICKUP SUMRY
TBRSAV	ODKINP RLAXIN SELECT
TD2OUT	DRIVER PRINT TBLEDG TD2P
TD2RE	PICKUP SUMRY
TEQEC	ODES
THISP	PICKUP SUMRY
THRUAT	NZGEOM PICKUP SUMRY TD2P

COMMON BLOCK	REFERRED TO BY SUBROUTINE
TOTSC	DRIVER IAUX MAIN1D
TRICOM	DRIVER SUMRY TRAJEC
TSTABL	DRIVER IAUX MAIN1D
TWOPHZ	CONVRT FLU MAIN1D ODK ODKINP OUTPUT PACK
VERSON	SUMRY
WHERE	CONTR1 CONTR2 FIG1 FIG2 PRISM1 PRISM2
WRODK	ODK OUTPUT
WTFLOW	BARPRO DIRECT READSI
ZDEBUG	DERIV STF
ZTRAN	DRIVER IAUX MAIN1D

COMMON BLOCK: BALCOM			
FORTRAN NAME	TYPE	DIMENSION	Description
TIM	R	150	Time, t_i, point from BAL module
PTC	R	150	total pressure at the end of the grain, P_{ci}, (PSIA)
RSTAR	R	150	r^*_i (ft)
XLST	R	150	L^*_i (in)
ETACS	R	150	η_{c^*i}
XMDOT	R	150	$\dot{m}_i$, mass flow rate (slugs/sec)
EMIEMP	R	150	not used
PCI	R	150	not used
XMTØT	R	150	not used
FID	R	150	not used
PMAX	R	-	maximum chamber pressure (PSIA)
NBPT	I	-	number of ballistics time points
PAVE	R	-	average P_c (PSIA)
RMIN	R	-	smallest r* in the RSTAR array (ft)
XLAVE	R	-	average L* (in)
ECAVE	R	-	average η_{c^*}
XMAVE	R	-	average $\dot{m}$ (slugs/sec)

Table 3-3 Common Block: BALCØM

COMMON EEE			
FØRTRAN NAME	TYPE	DIMENSION	DESCRIPTIØN
ETAC	R	15	calculated η
ETAE	R	15	empirical η
ETAI	R	15	input η
ETAU	R	15	value of η to be used
IPETA	I	15	pointer to value of η to be used

Table 3-4 Common Block EEE

COMMON BLOCK: LKEQKN			
FORTRAN NAME	TYPE	DIMENSION	DESCRIPTION
IPUISP	I	-	if IPUISP =1; store ϵ and ISP in AEQR and XIEQR
IREQ	I	-	if IREQ=1; restricted equilibrium option; if 0, regular equilibrium
AEQR	R	50	ϵ_i corresponding to I_{sp_i} (XIEQR) for the restricted equilibrium option
XIEQR	R	50	
NEQR	I	-	number of restricted equilibrium points
SOLPT	L	-	if TRUE solidification point properties calculated
FRZISP	R	-	value of frozen ISP

Table 3-5 Common Block LKEQKN

COMMON BLOCK: LKØDK			
FORTRAN NAME	TYPE	DIMENSION	Description
AKIN	R	40	ϵ_i corresponding to XIK(I)
XIK	R	40	kinetic I_{sp} calculated by ODK
ETAKIN	R	-	η_{KIN} at the maximum area ratio specified in the $GEOM namelist
NKPT	I	-	number of points in the AKIN, XIK arrays

Table 3-6 Common Block: LKØDK

CØMMØN BLØCK: LKTBL		
FØRTRAN NAME	TYPE	DESCRIPTION
IFØDE	I	if IFØDE=1, ODE data is available to the TBL module
IFTD2P	I	if IFTD2P=1, TD2P data is available to the TD2P module
INUN	I	logical unit on which the TD2P data is available

Table 3-7 Common Block LKTBL

CØMMØN BLØCK: MAN	
FØRTRAN NAME	DESCRIPTIØN
ØDF BAL ØDK TD2P TBL	if value = 0.0 not considered if value = 1.0 module by that name is to be executed if value = 2.0 module data was previously generated if value = 3.0 loss to be read in

Table 3-8 Common Block MAN

COMMON BLOCK: MARKET		
FORTRAN NAME	TYPE	DESCRIPTION
LODE	L	if .TRUE. module ODE is to be executed
LODK	L	if .TRUE. module ODK is to be executed
LTD2P	L	if .TRUE. module TD2P is to be executed
LTBL	L	if .TRUE. module TBL is to be executed
LBAL	L	if .TRUE. module BAL is to be executed

Table 3-9 Common Block MARKET

CØMMØN BLØC: TBLRE			
FORTRAN NAME	TYPE	DIMENSIØN	DESCRIPTIØN
NTBL	I	-	number of TBL points
ABL	R	150	A/A* from TBL
DELISP	R	150	$\Delta I_{sp_{TBL}}$
DISP	R	-	$\Delta I_{sp_{TBL}}$ at the maximum area ratio specified in the $GEOM namelist

Table 3-10 Common Block: TBLRE

COMMON BLOCK: TD2RE			
FORTRAN NAME	TYPE	DIMENSION	Description
A2D	R	150	ϵ_i corresponding to values of XI2D and E2D
XI2D	R	150	I_{sp} (ϵ_i) from TD2P
E2D	R	150	η'_{TD2P} (ϵ_i)
NTDPT	I	-	number of points in the A2D, XI2D and E2D array
ETD2P	R	-	time average value of η'_{TD2P}
XITD2P	R	-	time averaged value of $I_{sp_{TD2P}}$
ETD2PA	R	-	η_{TD2P} corrected for throat erosion
CD	R	-	nozzle discharge coefficient as calculated by TD2P
WDØT2D	R	-	nozzle mass flow rate as calculated by TD2P

Table 3-11 Common Block: TD2RE

COMMON THISP		
FØRTRAN NAME	TYPE	DESCRIPTION
ISP	R	Theoretical I_{sp} at maximum area ratio specified in the $GEOM namelist
CSTAR	R	Theoretical c* (ft/sec) as calculated by the ØDE module

Table 3-12 Common Block THISP

COMMON BLOCK: THROAT		
FORTRAN NAME	TYPE	DESCRIPTION
DRSDT	R	dr*/dt (ft/sec)
TBURN	R	burn time (sec)
NAR	I	number of points, ϵ_i at which nozzle erosion data is known
FSUBM	R	length of submergence/length of internal motor
AEAS	R	nozzle entrance area/nozzle throat area
AVELS	R	motor average L* (in)
KNØS	I	nozzle type flag; =1 steel nozzle, =0 all others
FCONP	R	mole fraction of condensed phase (moles/100 grs of product)

Table 3-13 Common Block THROAT

COMMON BLOCK: TRJCOM			
FORTRAN NAME	TYPE	DIMENSION	Description
ITJF	I	-	trajectory flag, if 0 no trajectory input
PAMBI	R	51	ambient pressure (PSIA)
TTRJ	R	51	times corresponding to PAMBI
NTJPT	I	-	number of trajectory points input

Table 3-14 Common Block TRJCOM

4.0 Program Files

The SPP program uses nine files to read, write, punch and temporarily save data. Six of these files are always referred to by variable names and hence, may be changed to suit the restrictions of the computer system being used. These variable names are set in Subroutine DRIVER for units 25 through 29 and by input data for unit 3 in subroutine NZGEØM.

Logical unit 8 is equated to the system punch file for CDC 6000 series computers and it is rewound. Therefore, care must be taken in assigning this file on machines which do not allow rewinds on the system punch file (e.g., Univac 1108). This linkage data written out on this unit is described in Volume III, Section 2.11 and will not be repeated here.

Table 4-1 lists the files used by the SPP code and the FORTRAN variable names used to reference these files. The device types listed correspond with the current usage on CDC 6000 series computers. The mnemoics used in Table 4-1 are listed below.

T - Tape files
MS - Mass Storage (perm) files
TMS - Temporary Mass Storage (scratch) files

FORTRAN VARIABLE NAME	Logical Unit	Device Type	Description
INP	3	T, MS, TMS	alternate input file for SPP linkage data
-	5	card reader	data input stream
-	6	line printer	printed output stream
-	8	punch	module linkage data output file
JANAF	25	T, MS, TMS	thermodynamic data file
KREAX	26	TMS	temporary file used to store reaction rate data
KSTF	27	TMS	temporary file used to store a short version of the thermodynamic data
IRREAD	28	TMS	temporary file used for multiple cases in the ØDK module.
ITSTAB	29	TMS	the data written on this file is not currently used

Table 4-1 Files Used By The SPP Code

5.0 PROGRAM SUBROUTINE DESCRIPTIONS

This section of Volume II contains a subroutine by subroutine description of the SPP computer code. Dictionaries of some of the more important variables which appear in labeled common areas of the program appear in Section 3.

Section 5 is organized in accordance with the overlay structure of the SPP code and the user is advised to refer to Section 3 of this document for the location of individual subroutines and common blocks and their cross references.

5.1 Link 00 Subroutines

5.1.1 Program Driver

This is the main routine for the SPP code and as such provides the overlay control, defines most of the upper level labeled common blocks, and initializes most of the control variables used in the code. On CDC 6000 computers this routine also defines all of the files and buffer sizes which are used in this code.

The DRIVER routine is essentially driven by directive cards which are read in on logical unit 5. The directive cards which are recognized by this subprogram are listed below in Table 5.1-1 along with the action taken by the DRIVER routine.

Table 5.1-1 Data Input Directives

Directive card	Action Taken
THERMØ	Call in overlay 1,0 which processes thermodynamic data.
TITLE	Stores the case title for subsequent print out.
PRØBLEM	Calls in the PRØBLM subroutine which determines which loss modules are to be executed. This directive and subsequent data initiate the execution of the various loss modules.
LOW T CPHS	Calls in subroutine LTCPHS which processes low temperature thermodynamic data.
GEØMETRY	Calls in subroutine GEØM which processes the nozzle geometry data.
TRAJECTØRY	Call in subroutine TPAJ which processes the trajectory and/or ambient pressure history of the motor.

In addition to processing the directive cards and calling in the selected loss modules, the main program also handles the communication of the linkage data between modules and calls the summary data output routine SUMRY.

5.1.2 Subroutine BLØCK DATA

BLØCK DATA contains atomic data stored in ATØM(i,j) and many of the variables used with the variable format, FMT. The ATØM variables are defined in appendix B, Reference 1. The format variables are stored in the common labeled ØUPT and are described here.

A variable format was used so that one format, FMT, could be used in the final output with changes in the number of decimal places according to the sizes of the numbers. The format is used to print a label and from 1 to 13 associated numbers. The labels contain 14 alphameric characters stored in four words and printed with 3A4, A2. The numbers are all printed in a field of 9. FMT is initially set in BLØCK DATA as follows:

FMT	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
	(1H	,3A4	,A2,	F9.	0,	F9.	0,	F9.	0,	F9.	0,
FMT	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	
	F9.	0,	F9.	0,	F9.	0,	F9.	0,	F9.	0,	
FMT	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)		
	F9.	0,	F9.	0,	F9.	0,	F9.	0	) .		

where the spaces are stored as blanks.

Some variables set in BLØCK DATA to modify FMT are as follows:

Variable:	F0	F1	F2	F3	F4	F5	FB	FMT13	FMT9X	FMT19
Storage:	0,	1,	2,	3,	4,	5,		13,	9X,	19,

The following is a list of variables used as labels and printed with 3A4, A2 in FMT:

Variable	Stored label
FP	P, ATM
FT	T, DEG K
FH	H, CAL G
FS	S, CAL/(G) (K)
FM	M, MOL WT
FV	(DLV/DLP) T
FD	(DLV/DLT) P
FC	CP, CAL/(G) (K)
FG	GAMMA (S)
FL	SON VEL, M/SEC
FR1	PC/P
FC1	CF
FN	MACH NUMBER
FR	CSTAR, FT/SEC
F1	ISP, LB-SEC/LB
FA	IVAC, LB-SEC/LB
FA1, FA2	AE/AT

5.1.3 Subroutine ATMØS

Soubroutine ATMØS is used by subroutine TRAJEC (see Section 5.1.16) to calculate the ambient pressure history during a motor firing if the ambient condition time history is given as a function of altitude.

This subprogram calculates, given the geometric altitude, the ambient temperature (°R), pressure (psia), sound speed (ft/sec), density (slugs/ft^3), and pressure gradient (lb/ft^3). Subroutine ATMØS assumes that the atmosphere consists of a gas of constant molecular weight on a spherical earth whose atmosphere is in a hydrostatic balance. The temperature profiles used in the hydrostatic balance are those given in the 1962 ARDC standard atmosphere tables.

These assumptions yield data which agree with that standard atmosphere table to within 1% up to altitudes of 700 kilometers (12,296,585 ft.).

5.1.4 Subroutine AVEBAL

Subroutine AVEBAL is called by subroutine PICKUP and calculates from the linkage data supplied by the internal ballistics calculation the time averaged values of chamber pressure, L*, throat radius, η_{C^*}, and mass flow rate. Also printed out are the maximum and minimum values of the above quantities along with their time histories. The throat erosion rate in inches/second is also calculated and printed out.

5.1.5 Subroutine ETACFX

This subroutine calculates the nozzle performance losses using empirical correlations. The losses calculated in this routine are in terms of per-cent nozzle thrust coefficient loss. Subroutine SUMRY converts these losses into fractional I_{sp} efficiencies using the equation

$$\eta_{ISP_i} = \frac{100-\bar{\eta}_i}{100}\ \eta_{c^*} \qquad 5.1\text{-}1$$

where $\bar{\eta}_i$ is the nozzle C_F loss as calculated below

Care should be taken when using the above equation that c* efficiency, η_{c^*}, is not used more than once to convert C_F efficiencies to I_{sp} efficiencies.

The losses considered and equations used by this routine are given below. A more complete description of these losses is given in Vol. I, Section 4.

Nozzle divergence loss is calculated as:

$$\bar{\eta}_{DIV} = 50\left[1 - \cos\left(\frac{\alpha+\theta_{EX}}{2}\right)\right] \qquad 5.1\text{-}2$$

where α = half angle of conical nozzle, included angle of contoured nozzle

θ_{EX} = exit angle of contoured nozzle

The loss due to finite rate chemical kinetics is calculated as:

$$\eta_{KIN} = 33.3\left[1 - \frac{\text{theoretical frozen Isp}}{\text{theoretical shifting Isp}}\right] \cdot \left(\frac{200}{P}\right) \qquad 5.1\text{-}3$$

where P=nozzle chamber pressure in psia or 200, whichever is greater.

The boundary layer loss is expressed in the form of a time-dependent heat transfer expression, with a correction term for nozzle expansion ratio:

$$\eta_{BL} = C_1 \frac{P^{0.8}}{D_t^{0.2}}\left[1 + 2\exp\left(-C_2 P^{0.8} t/D_t^{0.2}\right)\right]\left[1 + 0.016(\epsilon - 9)\right] \qquad 5.1\text{-}4$$

where P = pressure, psi

D_t = throat diameter, in

t = time, sec

ϵ = expansion ratio

The coefficients C_1 and C_2 are determined as is shown below. The nozzle type is input by the user as KNØZ in the $GEØM namelist.

Ordinary Nozzles: $C_1 = 0.00365$

$C_2 = 0.000937$

Steel Nozzles: $C_1 = 0.00506$

$C_2 = 0$

The nozzle two phase flow loss term is calculated as:

$$\eta_{TP} = C_3 \frac{\xi^{C_4} D_P^{C_5}}{P^{0.15} \epsilon^{0.08} D_t^{C_6}} \qquad 5.1\text{-}5$$

where ξ = mole fraction of condensed phase, moles/100 gm

D_P = particle size, μ

P = pressure, psi

ϵ = expansion ratio

D_t = throat diameter, in.

The coefficients are determined from the following:

If $\xi \geq 0.09$, $C_4 = 0.5$

$D_t < 1$:	$C_3 = 9.0$,	$C_5 = 1.0$,	$C_6 = 1.0$
$1 \leq D_t \leq 2$:	$C_3 = 9.0$,	$C_5 = 1.0$,	$C_6 = 0.8$
$D_t > 2$: and $D_P < 4$:	$C_3 = 13.4$,	$C_5 = 0.8$,	$C_6 = 0.8$
$4 \leq D_P \leq 8$:	$C_3 = 10.2$,	$C_5 = 0.8$,	$C_6 = 0.4$
$D_P > 8$:	$C_3 = 7.58$,	$C_5 = 0.8$,	$C_6 = 0.33$

If $\xi < 0.09$, $C_4 = 1.0$

$D_t < 1$:	$C_3 = 30.0$,	$C_5 = 1.0$,	$C_6 = 1.0$
$1 \leq D_t \leq 2$:	$C_3 = 30.0$,	$C_5 = 1.0$,	$C_6 = 0.8$
$D_t > 2$ and $D_P < 4$:	$C_3 = 44.6$,	$C_5 = 0.8$,	$C_6 = 0.8$
$4 \leq D_P \leq 8$:	$C_3 = 34.0$,	$C_5 = 0.8$,	$C_6 = 0.4$
$D_P > 8$:	$C_3 = 25.2$,	$C_5 = 0.8$,	$C_6 = 0.33$

with $D_p = 0.454\, P^{1/3}\, \xi^{1/3} \left[1 - \exp(-0.004L^*)\right]\left[1 + 0.045D_t\right]$ 5.1-6

and L* = motor characteristic length, in.

The motor submergence loss is calculated by:

$$\eta_{SUB} = 0.0684 \left(\frac{P\xi}{A^*}\right)^{0.8} \frac{S^{0.4}}{D_t^{0.2}} \qquad 5.1\text{-}7$$

where

P = pressure, psi

ξ = mole fraction of condensed phase, moles/100 gm

A* = nozzle entrance area/nozzle throat area

S = length of submergence/length of internal motor

D_t = throat diameter, in.

5.1.6 Subroutine FIND

This subroutine locates the indes, I, in a table such that $X(I) \leq X < X(I+1)$. On option, when the variable IEXTP in labeled common IXTRAP≠0, subroutine FIND allows for extrapolation in tables. See Appendix A for conversion aides.

5.1.7 Subroutine LINK1A

Subroutine LINK1A's only function is to call subroutine PRØBLM. It is provided to allow addition overlay capability on computers with explicid overlay calling features.

5.1.8 Subroutine LTCPHS

This subroutine processes the low temperature C_p, H, S Thermodynamic Data extension input as described in detail in Volume III, Section 2.2.1. The low temperature values of C_p, H, and S are at present only used in the ØDK module.

5.1.9 Subroutine NZGEØM

Subroutine NZGEØM is called by program DRIVER when a GEØMETRY directive is encountered in the card input stream. This routine uses the $GEØM Namelist input to define the nozzle geometry. Data which are read under this Namelist are stored in labeled common blocks LKGEØM, LKBM, ARPRNT, BALCØM, LKTD2P, and THRØAT.

The variables in the above common blocks are described in Section 3.2. However, for completeness, the variable names whose values are determined in subroutine NZGEØM are listed below in Table 5.1-1.

CØMMØN BLØCK	VARIABLE NAME
ARPRNT	ASUB ASUP NASUB NASUP
BALCØM	RSTAR TIM
LKBM	NNØZ R
LKGEØM	GMVAR IGM IWF NWS RS ZS
LKTD2P	RSTARM XLS
THRØAT	DRSDT FSUBM NAR TBURN

Table 5.1-1 Common Variables Set in NZGEØM

The variables shown in the above table are communicated downward only in the SPP code. That is, values set in this routine may be changed in individual calculational modules, however, the values which are changed are not transmitted to other calculation modules. Instead the initial values set via the $GEØM Namelist are used.

5.1.10 Subroutine PICKUP

Subroutine PICKUP processes all of the linkage data between the five basic computational modules which are transmitted on logical unit 8 except for the boundary layer edge conditions generated by the TD2P module for the TBL module.

This routine also performs unit conversion on the data read in, checks for overflow of tables, and prints out a summary of the results for each of the modules which have been executed.

5.1.11 Subroutine PRØBLM

Subroutine PRØBLM is executed by program DRIVER when the PROBLEM directive is encountered in the card input stream.

This routine sets flags and determines via user input in the $PRØB Namelist which of the five basic computational loss modules are to be executed. If the option to read previously generated data is selected, subroutine PICKUP is called by PRØBLM to read these data.

5.1.12 Subroutine SETUP

Subroutine SETUP is used to position the linkage data file, logical unit 8, for processing by subroutine PICKUP. This routine reads through the linkage data file until the appropriate calculation module name is found, then it backspaces one logical record so that the loss module name is the first card image read by subroutine PICKUP.

5.1.13 Subroutine SPLN

Performs either cubic or linear interpolation between two given points.

Cubic interpolation for a function and its first two derivatives is performed as described below: Given function values y_n and y_{n+1} and first derivative values y'_n and y'_{n+1} at x_n and x_{n+1}, this subroutine evaluates $y(x)$, $y'(x)$, and $y''(x)$ for $x_n \le x < x_{n+1}$ using:

$$y = A(x - x_n)^3 + B(x - x_n)^2 + C(x - x_n) + D$$

$$y' = y'_n + \frac{x - x_n}{x_{n+1} - x_n} \cdot \left[y'_{n+1} - y'_n \right]$$

$$y'' = (y'_{n+1} - y'_n) / h$$

where

$$A = \frac{1}{h^3} \cdot \left[(y'_{n+1} + y'_n)\, h - 2k \right]$$

$$B = -\frac{1}{h^3} \cdot \left[(y'_{n+1} + 2y'_n)h - 3k \right]$$

$$C = y'_n$$

$$D = y_n$$

$$h = x_{n+1} - x_n$$

$$k = y_{n+1} - y_n$$

Linear interpolation for a function and its first two derivatives is performed as described below:

$$y = y_n + \frac{x - x_n}{x_{n+1} - x_n} \cdot \left[y_{n+1} - y_n \right]$$

$$y' = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}$$

$$y'' = 0.0$$

5.1.14 Subroutine SUMRY

This subroutine is called by program DRIVER (Section 5.1.1) and prints out a summary report on the results of each of the five basic computational modules (if executed) at the maximum area ratio requested in the $GEØM namelist. If the GEØMETRY directive was not input to the SPP code the SUMRY routine assumes that one or more of the calculation modules were executed independently and that no summary of results is desired.

There are three basic sources for each of the losses considered by the SPP code. These sources are due to one of the following items shown in the table below.

1. User Input
2. Execution of a loss module
3. Calculation of an empirical correlation

Table 5.1-3 Sources of Losses Considered

User input of loss efficiencies along with selection of which of the loss modules are to be executed is determined by input to subroutine PRØBLM (Section 5.1.11). Empirical losses are calculated by subroutine ETACFX (Section 5.1.5) which is called by SUMRY. While all of the above losses are printed out in this routine, the selection of which source of the loss is to be used in determining the total delivered performance of the solid propellant rocket motor is given in the order appearing in Table 5.1-3. That is, if the user of the SPP code inputs a loss, that is the loss efficiency that is used in calculating the final delivered performance. Hence, while user input overrides values calculated by the analytical loss modules, the values calculated by the loss modules takes precedence over the values determined by the empirical correlations.

Using the selection criteria described above, subroutine SUMRY then prints out the product of efficiencies along with the decrement of performance due to a turbulent boundary layer. Hence, if no input efficiencies were input, the value of the delivered performance due to input losses would then be just the theoretical I_{sp} calculated by the ØDE module.

In order to determine the motor delivered performance to ambient conditions, trapezodial rule integration is used to calculate the average ambient pressure and average expansion ratio if this information is available. That is, the average performance delivered to ambient conditions is calculated by

$$\bar{I}_{sp_D} = I_{sp_{th}} \prod_{i=1}^{n} \eta_{i,k} - \Delta I_{sp_{TBL}} - \Delta I_{sp_A} \qquad 5.1\text{-}8$$

where $I_{sp_{th}} = I_{sp_{VAC}}$ calculated by ØDE

$\eta_{i,k}$ = fractional loss efficiency determined via the selection rule, k, for the i^{th} loss considered

$\Delta I_{sp_{TBL}}$ = loss decrement due to the formation of a turbulent boundary layer

ΔI_{sp_A} = loss decrement due to the motor delivering thrust to an ambient pressure. This value is calculated as $(\bar{P}_{amb}/\bar{P}_c) \times \bar{\epsilon} \times c^* / (g_c \times C_D)$

If both the ballistics and the two dimensional-two phase flow modules linkage data are available, then subroutine SUMRY calculates the delivered performance throughout the motor firing for each time point specified in the trajectory and ballistics input data. The following formulas are used to compute the values printed out by this routine.

$$\eta_{CE}(t) = \max(1.0, C_D * \eta_{c^*}(t))$$

$$I_{sp_{VAC}}(t) = I_{sp_{th}} \cdot \eta'_{TD2P}(t) \cdot \left(\frac{I_{sp_{TD2P}}(t)}{I_{sp_{TD2P}}(t=0)}\right) \cdot \eta_{CE}(t) \cdot \eta_{SUB} \cdot \eta_{KIN} - \Delta I_{sp_{TBL}}$$

$$F_{VAC}(t) = I_{sp_{VAC}} * \dot{m}(t)$$

$$F_{amb}(t) = P_{amb}(t) \cdot A_{exit}$$

$$I_{sp_{amb}}(t) = F_{amb}(t)/\dot{m}(t)$$

$$F_{DEL} = F_{VAC}(t) - F_{amb}(t)$$

$$I_{sp_{DEL}} = I_{sp_{VAC}} - I_{sp_{amb}}$$

$$I = \int_0^t F_{DEL}(t)\, dt$$

$$m = \int_0^t \dot{m}\, dt$$

where

$\dot{m}$ = mass flow rate (from Ballistics calculation)

η_{c^*} = c* efficiency (from Ballistics calculation)

P_{amb} = ambient pressure (from input tables)

η_{SUB} = submergence loss efficiency (selected as described above)

η_{KIN} = kinetic loss efficiency (selected as described above)

$\Delta I_{sp_{BL}}$ = boundary layer loss decrement (selected as described above)

η'_{TD2P} = 2D-2phase flow loss efficiency (from TD2P as a function of $\epsilon(t)$)

$I_{sp_{TD2P}}$ = 2D-2phase I_{sp} (from TD2P as a function of $\epsilon(t)$)

A_{exit} = nozzle exit area (from input)

F_{amb} = thrust decrement due to ambient pressure

I = total impulse to this time

m = total mass expended to this time

$I_{sp_{DEL}}$ = delivered specific impulse

F_{DEL} = delivered thrust

$I_{sp_{amb}}$ = I_{sp} decrement due to ambient pressure

5.1.15 Subroutine TIMERX

This subroutine is provided to localize printing of computer execution times during the calculations. This subroutine should be modified at each local installation to provide proper interface with local system timing routines. This subroutine calls a CDC system subroutine named SECØND to obtain the current run execution time in seconds.

5.1.16 Subroutine TRAJEC

Subroutine TRAJEC is used to calculate ambient pressure during a motor firing. A table of ambient pressure or altitude is input to this routine via the Namelist name $TRAJ as a function of time. Delivered performance to the specified ambient conditions is then calculated in subroutine SUMRY and printed. If an altitude versus time table is input to subroutine TRAJEC, then subroutine ATMØS is used to calculate the ambient pressure at which the performance of the motor is delivered.

5.2 Link 10 Subroutines

5.2.1 Subroutine LINK10

This subroutine consists only of a call to subroutine TTAPE and is used only to facilitate conversion of the program overlay structure to other computers.

5.2.2 Subroutine TTAPE

When a THERMØ directive card is read by the main program this subroutine is called to generate a master Thermodynamic Data tape (Logical Unit 25). The input Thermodynamic Data is in curve fit form and is identical to that required for the ØDE computer program described in NASA SP-273, Reference 1. The format for this curve fit data is described in the Volume III, Section 2.2. This routine does not update the master Thermodynamic Data tape, but instead creates a new one. Hence, all of the thermodynamic must be read in, old and new, if the user wishes to add new species to this file.

5.3 Link 20 Subroutines

Overlay Link 20 contains the ØDE module which calculates the theoretical or reference I_{sp} used in the methodology incorporated in the SPP code.

The main body of subprograms which are contained in this overlay are from the NASA-Lewis Equilibrium Program (Ref.1) as modified for rocket performance calculations (Ref. 3). The brief analytical descriptions of the equilibrium calculations contained herein were condensed from Reference 1. For a complete description of these calculations see Reference 1 (NASA SP-273).

5.3.1 Subroutine LINK20

This subroutine consists only of a call to subroutine ØDES and is used only to facilitate conversion of the program overlay structure to other computers.

5.3.2 Subroutine CPHS

This subroutine evaluates the thermodynamic functions $\frac{Cp^\circ}{R}$, $\frac{H^\circ_T}{RT}$, $\frac{S^\circ_T}{R}$ from curve fit coefficients. Two sets of coefficients are used for two adjacent temperature intervals. The functions evaluated are presented below:

$$\frac{Cp^\circ_T}{R} = a_1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4$$

$$\frac{H^\circ_T}{RT} = a_1 + \frac{a_2T}{2} + \frac{a_3T^2}{3} + \frac{a_4T^3}{4} + \frac{a_5T^4}{5} + \frac{a_6}{T}$$

$$\frac{S^\circ_T}{R} = a_1 \ln T + a_2T + \frac{a_3T^2}{2} + \frac{a_4T^3}{3} + \frac{a_5T^4}{4} + a_7$$

$$\frac{G^\circ_T}{RT} = \frac{H^\circ_T}{RT} - \frac{S^\circ_T}{R}$$

5.3.3 Subroutine DETON

This is a dummy subroutine used to replace subroutine DETØN of Reference 1.

5.3.4 Subroutine EFMT

Subroutine EFMT (E-format) writes statements in a special exponent form. This form is similar to the standard FORTRAN E-format, but the letter E and some of the spaces have been removed for compactness. This routine is not used in the SPP code at the present time.

5.3.5 Subroutine EQLBRM

EQLBRM is the control routine for the equilibrium module which calculates equilibrium compositions and thermodynamic properties for a particular point. A free-energy minimization technique is used. This routine permits calculations such as (1) chemical equilibrium for assigned thermodynamic states (T,P), (H,P), (S,P), (T,V), (U,V), or (S,V), (2) theoretical rocket performance for both equilibirum and frozen compositions during expansion, (3) incident and reflected shock properties, and (4) Chapman-Jouguet detonation properties. The program considers condensed species as well as gaseous species. A detailed description of the equations and computer program for computations involving chemical equilibria in complex systems is given in Reference 1. Figures 4(a) through 4(c) of Reference 1 give a complete flow diagram for this subroutine.

5.3.6 Subroutine FRØZEN

Subroutine FRØZEN is called from subroutine RØCKET to calculate the temperature and thermodynamic properties for the following assigned conditions:

1. Composition frozen at combustion conditions.
2. At assigned exit pressure.
3. At assigned entropy equal to the entropy at combustion conditions.

The iteration procedure used for obtaining the exit temperature is discussed in the section Procedure for Obtaining Frozen Rocket Performance (p. 40, Reference 1).

This subroutine has been modified to calculate the strictly frozen performance at well below the solidification point of any condensed phases which are present.

5.3.7 Subroutine GAUSS

Subroutine GAUSS is used to solve the set of simultaneous linear iteration equations constructed by subroutine MATRIX. The solution is effected by performing a Gauss reduction using a modified pivot technique. In this modified pivot technique only rows are interchanged. The row to be used for the elimination of a variable is selected on the basis that the largest of its elements, after division by the leading element, must be smaller than the largest element of the other rows after division by their leading elements.

The solution vector is stored in X(k). In the event of a singularity, IMAT (which is equal to the number of rows) is set equal to IMAT - 1. IMAT is tested later in subroutine EQLBRM.

5.3.8 Subroutine HCALC

The purpose of HCALC is to calculate thermodynamic properties for reactants under certain circumstances. HCALC is called from entry NEWØF of SAVE.

HCALC is called from NEWØF when CALCH is set true. CALCH is set true in the main program when zeros have been punched in card columns 37 and 38 on one or more REACTANTS cards. The zeros are a code indicating that the enthalpy (or internal energy for UV problems) for the reactant should be calculated from the THERMØ data at the temperature punched on the card. This temperature has been stored in the RTEMP array. CPHS is called to calculate the enthalpy. The value is stored in the ENTH array and printed in the final tables.

The properties calculated in subroutine HCALC, their FORTRAN symbols, and the condition for which they are used are as follows:

Property	FØRTRAN Symbol	When used
H(K)T	HPP(K)	if cc 37 and 38 are
h_o/R	HSUBO	input as 00 on reactants card

5.3.9 Subroutine MATCH

This subroutine is called by RØCKET to supply subroutine MUK with a vector of internal sequence numbers which point to the appropriate Lennard-Jones parameters used by MUK to calculate transport properties. The input to MATCH is the species name (from the SUB array) and the output corresponds to the index numbers in the table in the MUK write up.

5.3.10 Subroutine MATRIX

This subroutine sets up the matrices corresponding to tables I through IV of Reference 1. The assigned thermodynamic state being set up (tables I and II) is specified by the following codes:

Assigned thermodynamic state	Codes
TP	TP = .TRUE. VOL = .FALSE. CONVG = .FALSE.
HP	HP= .TRUE. VOL = .FALSE. CONVG = .FALSE.
SP	SP = .TRUE. VOL = .FALSE. CONVG = .FALSE.
TV	TP = .TRUE. VOL = .TRUE. CONVG = .FALSE.
UV	HP= .TRUE. VOL = .TRUE. CONVG = .FALSE.
SV	SP = .TRUE. VOL = .TRUE. CONVG = .FALSE.

After convergence of any of the previous six problems, setup of the derivative matrices (tables III and IV) is specified by the following codes:

Derivative	Codes
DLVTP	CONVG = .TRUE. LOGV = .FALSE.
DLVPT	CONVG = .TRUE. LOGV = .FALSE.

Note: Only the Rocket option is available in the SPP code.

5.3.11 Subroutine MUK (CP, XW, T, C, IT, N, XM, XMT, TK, PR)

This routine calculates the viscosity, thermal conductivity and Prandtl number for a gas composed of a mixture of species.

CALLING SEQUENCE:

CP	is an array of N specific heats of the species ($ft^2/sec^2\ ^\circ R$)	(INPUT)
XW	is an array of N molecular weights of the species (slug/slug mole)	(INPUT)
T	is the temperature of the gas ($^\circ R$)	(INPUT)
C	is an array of N mass fractions of the species	(INPUT)
IT	is an array of N indices of the species into a table of collision diameter (σ) and energy of attraction (ϵ/k)	(INPUT)
N	is the number of species in the gas	(INPUT)
XM	is an array of N viscosities of the species, ($lbf \cdot sec/ft^2$)	(ØUTPUT)
XMT	is the total viscosity of the gas, ($lbf \cdot sec/ft^2$)	(ØUTPUT)
TK	is the thermal conductivity of the gas, ($ft \cdot lbf/ft^2\ sec\ ^\circ R/ft$))	(ØUTPUT)
PR	is the Prandtl number of the gas,	(ØUTPUT)

Method:

$$C_p = \sum_{i=1}^{N} C_i C_{p_i}$$ Specific Heat

$$M_w = \frac{1}{\sum_{i=1}^{N} \frac{C_i}{M_{w_i}}}$$ Molecular Weight

$$X_i = C_i \cdot \frac{M_w}{M_{w_i}}$$

$$T_i^* = \frac{T/1.8}{(\epsilon/k)_i}$$

$$\Omega_i = \text{table } (T_i^*)$$ table of Ω vs T^*

$$\sigma_i = \text{table } (i)$$ table of σ and ϵ/k vs. individual species

$$\mu_i = \frac{4.15822 \times 10^{-8} \sqrt{M_{w_i} T}}{\sigma_i^2 \Omega_i}$$

$$\Phi_{i_j} = \frac{1}{2^{3/2}} \left(1 + \frac{M_{w_i}}{M_{w_j}}\right)^{-1/2} \left[1 + \left(\frac{\mu_i}{\mu_j}\right)^{1/2} \left(\frac{M_{w_j}}{M_{w_i}}\right)^{1/4}\right]^2$$

$$\mu = \sum_{i=1}^{N} \left[\mu_i \left(1 + \sum_{\substack{j=1 \\ j \neq i}}^{N} \Phi_{i_j} \frac{X_j}{X_i}\right)^{-1}\right]$$ viscosity of the gas

$$K_i = \frac{\mu_i \, \Re}{M_{w_i}} \left(.45 + 1.32 \frac{C_{p_i}}{(\Re/M_{w_i})}\right)$$

$$K = \left[\sum_{i=1}^{N} K_i \left(1 + 1.065 \sum_{\substack{j=1 \\ j \neq i}}^{N} \Phi_{i_j} \frac{X_j}{X_i}\right)^{-1}\right]$$ thermal conductivity

$$P_r = \frac{C_p \mu}{K}$$ Prandtl number

Equations for K_i and μ_i are from Reference 7. The values of the collision integral are from table 2 of Appendix B in Reference 8. The relations used to calculate μ and K of the mixture are from Reference 9.

Also from Reference 7 are the values of the collision diameters, σ, and energy of attraction, ϵ/k.

Table 1 correlates the chemical name of the species to the internal number assigned to it by the subroutine. Also included in Table 1 is the key-punch name assigned to the species, since lower-case letters and subscripts are non-standard features in most computer configurations.

Table 1. SPECIES NAMES AND IDENTIFIERS

Species Number	Chemical Name	Key Punch Name
1	Al	AL
2	AlCl	ALCL
3	$AlCl_3$	ALCL3
4	AlF	ALF
5	AlF_3	ALF3
6	AlN	ALN
7	AlO	ALO
8	AlS	ALS
9	Al_2	AL2
10	Alr	AIR
11	Ar	AR
12	AsH_3	ASH3
13	B	B
14	BBr_3	BBR3
15	BCl	BCL
16	BCl_2	BCL2
17	BCl_3	BCL3
18	BF	BF
19	BF_2	BF2
20	BF_3	BF3
21	BI_3	BI3
22	BO	BO
23	$B(OCH_3)_3$	B(OCH3)3
24	B_2	B2
25	B_2H_6	B2H6
26	B_2O_3	B2O3
27	Be	BE
28	$BeBr_2$	BEBR2
29	BeCl	BECL
30	$BeCl_2$	BECL2
31	BeF	BEF
32	BeF_2	BEF2
33	BeI_2	BEI2
34	Be_2	BE2

Species Number	Chemical Name	Key Punch Name
35	Br	BR
36	BrF	BRF
37	BrF_3	BRF3
38	BrO	BRO
39	Br_2	BR2
40	C	C
41	$CBrF_3$	CBRF3
42	CBr_4	CBR4
43	CCl	CCL
44	$CClF_3$	CCLF3
45	CCl_2	CCL2
46	CCl_2F_2	CCL2F2
47	CCl_3	CCL3
48	CCl_3F	CCL3F
49	CCl_4	CCL4
40	CF	CF
51	CF_2	CF2
52	CF_3	CF3
53	CF_4	CF4
54	CH	CH
55	CHBrClF	CHBRCLF
56	$CHBrCl_2$	CHBRCL2
57	$CHBr_3$	CHBR3
58	$CHClF_2$	CHCLF2
59	$CHCl_3$	CHCL3
60	CHF_3	CHF3
61	CH_2BrCl	CH2BRCL
62	CH_2ClF	CH2CLF
63	CH_2Cl_2	CH2CL2
64	CH_2F_2	CH2F2
65	CH_2I_2	CH2I2
66	CH_3Br	CH3BR
67	CH_3Cl	CH3CL

Species Number	Chemical Name	Key Punch Name
68	CH_3F	CH3F
69	CH_3I	CH3I
70	CH_3OH	CH3OH
71	CH_4	CH4
72	CN	CN
73	CO	CO
74	COS	COS
75	CO_2	CO2
76	CP	CP
77	CS	CS
78	CS_2	CS2
79	C_2	C2
80	C_2H_2	C2H2
81	C_2H_4	C2H4
82	C_2H_6	C2H6
83	C_2H_5Cl	C2H5CL
84	C_2H_5OH	C2H5OH
85	C_2N_2	C2N2
86	CH_3OCH_3	CH3OCH3
87	CH_2CHCH_3	CH2CHCH3
88	CH_3CCH	CH3CCH
89	cyclo-C_3H_6	CYCLO-C3H6
90	C_3H_8	C3H8
91	n-C_3H_7OH	N-C3H7OH
92	CH_3COCH_3	CH3COCH3
93	CH_3COOCH_3	CH3COOCH3
94	n-C_4H_{10}	N-C4H10
95	iso-C_4H_{10}	ISO-C4H10
96	$C_2H_5OC_2H_5$	C2H5OC2H5
97	$CH_3COOC_2H_5$	CH3COOC2H5
98	n-C_5H_{12}	N-C5H12
99	$C(CH_3)_4$	C(CH3)4
100	C_6H_6	C6H6

Species Number	Chemical Name	Key Punch Name
101	C_6H_{12}	C6H12
102	n-C_6H_{14}	N-C6H14
103	Cd	CD
104	Cl	CL
105	ClCN	CLCN
106	ClF	CLF
107	ClF_3	CLF3
108	ClO	CLO
109	Cl_2	CL2
110	F	F
111	FCN	FCN
112	F_2	F2
113	H	H
114	HBr	HBR
115	HCN	HCN
116	HCl	HCL
117	HF	HF
118	HI	HI
119	HS	HS
120	H_2	H2
121	H_2O	H2O
122	H_2O_2	H2O2
123	H_2S	H2S
124	He	HE
125	Hg	HG
126	$HgBr_2$	HGBR2
127	$HgCl_2$	HGCL2
128	HgI_2	HGI2
129	I	I
130	ICl	ICL
131	I_2	I2
132	Kr	KR
133	Li	LI
134	LiBr	LIBR

Species Number	Chemical Name	Key Punch Name
135	LiCN	LICN
136	LiCl	LICL
137	LiF	LIF
138	LiI	LII
139	LiO	LIO
140	Li_2	LI2
141	Li_2O	LI2O
142	Mg	MG
143	MgCl	MGCL
144	$MgCl_2$	MGCL2
145	MgF	MGF
146	MgF_2	MGF2
147	Mg_2	MG2
148	N	N
149	NF_3	NF3
150	NH	NH
151	NH_3	NH3
152	NO	NO
153	NOCl	NOCL
154	N_2	N2
155	N_2O	N2O
156	Na	NA
157	NaBr	NABR
158	NaCN	NACN
159	NaCl	NACL
160	NaF	NAF
161	NaI	NAI
162	NaO	NAO
163	NaOH	NAOH
164	Na_2	NA2
165	Na_2O	NA2O
166	Ne	NE
167	O	O
168	OF	OF

Species Number	Chemical Name	Key Punch Name
169	OF_2	OF2
170	OH	OH
171	O_2	O2
172	P	P
173	PCl	PCL
174	PCl_3	PCL3
175	PF	PF
176	PF_3	PF3
177	PH_3	PH3
178	PN	PN
179	PO	PO
180	PS	PS
181	P_2	P2
182	P_4	P4
183	S	S
184	SF_6	SF6
185	SO	SO
186	SO_2	SO2
187	S_2	S2
188	S_2F_2	S2F2
189	Si	SI
190	SiCl	SICL
191	$SiCl_4$	SICL4
192	SiF	SIF
193	$SiFCl_3$	SIFCL3
194	SiF_2Cl_2	SIF2CL2
195	SiF_3Cl	SIF3CL
196	SiF_4	SIF4
197	SiH_4	SIH4
198	SiO	SIO
199	SiO_2	SIO2
200	SiS	SIS
201	Si_2	SI2
202	$SnBr_2$	SNBR2
203	$SnCl_4$	SNCL4
204	UF_6	UF6
205	Xe	XE
206	Zn	ZN

5.3.12 Subroutine ØDES

This is the main program for ØDE and corresponds to the Main Program described in Reference 1. Generally, the routine performs the following functions:

1. Reads code cards THERMØ, REACTANTS, ØMIT, INSERT, and NAMELISTS and directs flow of program accordingly.
2. Stores THERMØ data on tape.
3. Calls subroutine REACT to read and process REACTANTS cards.
4. Reads ØMIT and INSERT cards and stores species names.
5. Initializes variables in namelist $ØDE.
6. Reads and writes namelist $ØDE.
7. Converts assigned densities, if any, (RHO(i) in $ØDE) to specific volumes: VLM(i) = 1/RHO(i).
8. Stores the number of pressures or volumes in NP.
9. Stores values of o/f in ØXF array. If o/f values have not been input directly, they are calculated as follows:

Values	Code	o/f calculation in main program
Oxidant to fuel weight ratio, o/f	OF = .TRUE.	ØXF(i) = o/f
Fuel to air weight ratio, f/a	FA = .TRUE.	ØXF(i) = 1/(f/a)
Percent fuel, %F	FPCT = .TRUE.	ØXF(i) = (100-%F)/(%F)
Equivalence ratio, r	ERATIØ = .TRUE.	$\text{ØXF(i)} = \frac{-rV^{-}(2)_{-V^{+}(2)}}{rV^{-}(1)_{+V^{+}(1)}}$
Not specified		$\text{ØXF(i)} = \frac{WP(1)}{WP(2)}$

Values of WP(1) and WP(2) are defined in appendix B or Reference 1.

10. Makes necessary adjustments to consider charge balance if IØNS= .TRUE.. This is done by adding 1 to NLM and E to LLMT array.
11. Calls SEARCH to pull required THERMØ data from tape and to store the data in core.
12. Sets initial estimates for compositions. These estimates are set with each $ØDE read. They are used only for the first point in the lists of variables in namelist (e.g., the first o/f and the first T and P in a TP problem). All succeeding points use results from a previous point for estimates.

For the first point the program assigns an estimate of 0.1 for n, the total number of kilogram-moles per kilogram. The initial estimate of number of moles of each gaseous species per kilogram of mixture n_j is set equal to 0.1/m where m is the total number of gaseous species. Condensed species are assigned zero moles.

13. Sets IUSE(j) positive for condensed species listed on INSERT cards (see IUSE array).
14. Calls RØCKET is RKT is true.

Note: while the use of oxidizer to fuel ratio, Ø/F, is not generally used in describing solid rocket motor propellants, it can be used to great advantage in specifying binder, metal loading, and oxidizer ratios in parametric studies.

5.3.13 Subroutine OMEGA

This subroutine calculates the exponent, ω, used in the viscosity-temperature relationship

$$\mu = \mu_{ref} \left(\frac{T}{T_{ref}} \right)^{\omega}$$

using the method of least squares. That is, it calculates the value of ω which gives the smallest sum of the errors squared. This form of the viscosity-temperature relationship was selected since both the TD2P and TBL modules require viscosity data in this manner.

In order to supply the maximum amount of accuracy and also to minimize the variation in data due to the selection of an exit area ratio, it was decided to match the throat value of viscosity exactly and select an ω which would provide the best fit for viscosity at the chamber and exit of the motor.

The form of the error, E, was taken to be

$$E = \ln \mu/\mu^* - \omega \ln T/T^*$$

Squaring the errors, differentiating with respect to ω, and setting the results equal to zero, yields the following value for ω

$$\omega = (\ln T_c/T^* \ln \mu_c/\mu^* + \ln T_e/T^* \ln \mu_e/\mu^*)/(\ln T_c/T^*)^2 + (\ln T_e/T^*)^2)$$

where

T = temperature

μ = viscosity

e = refers to the exit plane

c = refers to the chamber

* = refers to the throat plane

5.3.14 Subroutine ØUT1

This subroutine, together with entries ØUT2 and ØUT3, writes statements common to all problems. ØUT1 writes statements giving the data on REACTANTS and on o/f, percent fuel, equivalence ratio, and density.

Entry ØUT2. - This entry writes the statements for printing values of pressure, temperature, density, enthalpy, entropy, molecular weight, $(\partial \ln V/\partial \ln P)_T$ (if equilibrium), $(\partial \ln V/\partial \ln T)_P$ (if equilibrium), heat capacity, γ_S, and sonic velocity. These variables and corresponding labels are printed with a variable format described in BLØCK DATA.

Entry ØUT3. - Entry ØUT3 writes statements giving the equilibrium mole fractions of reaction species.

Subroutine ØUT1 has also been modified to print out the following additional items:

a) gas velocity
b) molecular weight of the gas-condensed phase combined
c) mass fraction of condensibles
d) molecular weight of the condensibles in the chamber
e) molecular weight of the gas in the chamber
f) the erosion parameter, β

In addition, ØUT1 also traps the starting conditions for the ØDK module.

5.3.15 Subroutine REACT

The purpose of subroutine REACT is to read and process the data on the REACTANTS cards. The subroutine is called from the main program after a REACTANTS code card has been read. The data on these cards are described in the REACTANTS Cards section (p. 62) of Reference 1 and in Volume III, Section 2.6.1. References to page numbers and equations given below also pertain to Reference 1.

The reactants may be divided into two groups according to card column 72 on the REACTANTS cards. The two groups are oxidants (O in cc 72) and fuels (cc 72 $\neq$ O). We generally keypunch F in card column 72 for fuels even though this is not necessary. The contents of card column 72 are read into FØX. Depending on the contents of FØX, program variables relating to oxidants or fuels are subscripted 1 for oxidants and 2 for fuels.

The FORTRAN symbols for the properties read from the REACTANTS cards and their associated properties (discussed in INPUT CALCULATIONS, p. 55 of Reference 1) are as follows:

Property	FORTRAN symbol
$a_{ij}^{(k)}$	ANUM(j,m)[a]
$W_j^{(k)}$	PECWT(j) (if no M in cc 53)
$N_j^{(k)}$	PECWT(j) (if M in cc 53)
$(H^{\circ}_{T_j})^{(k)}$	ENTH(j) (if not UV Problem and 00 not in cc 37 and 38)
$(U^{\circ}_{T_j})^{(k)}$	ENTH(j) (if UV problem and 00 not in cc 37 and 38)
$\rho_j^{(k)}$	DENS(j)

[a]Each of the j REACTANTS cards contains from 1 to 5 stoichiometric coefficients read (indicated by subscript m) into ANUM(j,m) and their corresponding chemical symbols read into NAME(j,m). In relating an ANUM(j,m) with $a_{ij}^{(k)}$, the index i associated with a particular chemical element is determined from the chemical symbol in NAME(j,m).

If there are several oxidants their properties are combined by subroutine REACT into properties of a total oxidant using the relative proportion of each oxidant given on the REACTANTS cards. Similarly, if there are several fuels, their properties are combined into properties of a total fuel. The total oxidant and total fuel properties discussed in INPUT CALCULATIONS[1] and their associated FORTRAN symbols are as follows:

Property	FORTRAN symbol	Equation (see Ref. 1)
$b_i^{(k)}$	BOP(i,k)	(187)
$M_j^{(k)}$	RMW(j)	(190)
$\nu_T^{(k)}$	HPP(k) (if not UV problem and 00 not in cc 37 and 38)	(192)
$\upsilon_T^{(k)}$	HPP(k) (if UV problem and 00 not in cc 37 and 38)	(194)
$M^{(k)}$	AM(k)	(196)
$\rho^{(k)}$	RH(k)	(198)
$V^{+(k)}$	VPLS(k)	(200)
$V^{-(k)}$	VMIN(k)	(201)
$W_j^{(k)} \Big/ \sum_{j=1}^{NREAC} W_j^{(k)}$	PECWT(j)	

If any of the $\rho_j^{(k)}$ are zero then RH(1) = RH(2) = 0.

These total oxidant and total fuel properties are subsequently combined into total reactant properties by using the values of oxidant-fuel mixture ratios obtained from the main program. This is done in NEWØF, an entry point in SAVE.

Other common variables set by REACT are LLMT, NAME, ANUM, ENTH, FAZ, RTEMP, FØX, DENS, RMW, MØLES, NLM, NEWR, and NREAC.

A provision is made for eliminating a second tape search when two consecutive sets of REACTANTS cards contain the same elements. This is done by saving the element symbols (LLMT(ℓ)) in LLMTS(ℓ), the kilogram-atoms per kilogram (BOP(ℓ,k)) in SBOP(ℓ,k), and the number of elements (NLM) in NLS.

Atomic weights M_i used in equation (190)[1] are stored in ATØM(2,i). The corresponding chemical symbols are stored in ATØM(1,i). The oxidation states of the chemical elements V_i^+ or V_i^- used in equations (200) and (201)[1] are stored in ATØM(3,i). The ATØM array is stored by BLØCK DATA.

5.3.16 Subroutine RKTØUT

This subroutine calculates various rocket performance parameters from previously calculated thermodynamic properties.

It is also the control program for writing rocket performance output. It contains the WRITE statements that apply specifically to rocket parameters and it calls subroutine ØUT1 and entries ØUT2 and ØUT3 for the WRITE statements common to all problems. The rocket parameters are printed with the variable format, FMT, described in BLØCK DATA.

Subroutine RKTØUT is called from subroutine RØCKET.

5.3.17 Subroutine RØCKET

This subroutine is the control program for the RKT problem (rocket performance calculations discussed in section RØCKET PERFORMANCE).[1] A flow diagram for this subroutine is given in Figure 5 of Reference 1. Subroutine RØCKET obtains the required thermodynamic properties for equilibrium performance by calling subroutine EQLBRM. For forzen performance, subroutine RØCKET calls subroutine FRØZEN to obtain the required thermodynamic properties. Rocket performance parameters are then obtained by calling subroutine RKTØUT. In addition to calling RKTØUT, and FRØZEN, and in addition to using controls common to all problems (discussed in section MODULAR FORM OF THE PROGRAM, p. 75, Reference 1) subroutine RØCKET also does the following:

1. It calculates estimates for throat pressure ratios.
2. It calculates estimates for pressure ratios corresponding to assigned area ratios (if any).

Subroutine RØCKET was modified to perform the following tasks:

a) compute the area ratio at which solidification of condensed phases would occur if condensed phases are present.

b) on option, restricts the total amount of condensed phases during the expansion to the amount initially in the rocket motor chamber.

c) computes the print out station at which subroutine OUT1 will trap the start conditions for a kinetics, ØDK, calculation.

d) computes, using subroutiens MATCH, MUK, and ØMEGA, transport properties for the BAL, TD2P, and TBL modules.

e) writes (punches) out the linkage data necessary for communications with the various loss modules.

5.3.18 Subroutine SAVE

This subroutine has several functions, all of which are concerned with saving some information from a completed calculation for subsequent use in later calculations. The primary purpose is to save computer time by having good initial estimates for compositions.

These estimates for the next point, NPT, come from either the point just completed, ISV, or some other previous point. The flow of the routine is directed by ISV as follows:

1. ISV positive. Transfer compositions from the point just completed for use as initial estimates for next point (transfer EN(j, ISV) to EN(j, NPT)).
2. ISV negative. Save values of ENLN(j) for gases and EN(j) for condensed in SLN(j), ENN in ENSAVE, ENNL in ENLSAV, IQ1 in IQSAVE, JSØL in JSØLS, JLIQ in LKIQS, and NLM in LL1. (These values are saved because they are to be used as initial estimates for some future point and they may be overwritten in the meantime.) Make ISV positive and transfer EN(j,ISV) to EN(j,NPT).
3. ISV zero. Use the data previously saved (as discussed in 2. as initial estimates for current point. Restore IUSE codes and inclusion or exclusion of "E" as an element for IØNS option.

Entry NEWØF. - NEWØF combines the properties of total oxidant and total fuel calculated in subroutine REACT with an o/f value to give properties for the total reactant. NEWØF is called for each mixture assigned in the MIX array in $ØDE namelist. It is called from subroutine RØCKET. The calculated properties and corresponding FORTRAN symbols are as follows:

Property	FORTRAN symbol	Equation (See Ref. 1)
b_i^o	B0(i)	(191)
h_o/R	HSUB0 (if not UV problem)	(193)
u'_o/R	HSUB0 (if UV problem)	(195)
ρ_o	RHØP	(199)
r	EQRAT	(204)

Subroutine HCALC is called by Entry NEWØF to calculate the enthalpies for each reactant that has zeros keypunched in card columns 37 and 38 in its REACTANTS card.

Values of HPP(2), HPP(1), HSUB0, B0P(i,2), B0P(i,1), and B0(i) are printed out.

5.3.19 Subroutine SEARCH

This subroutine selects the Thermodynamic Data to be used in the problem. A scan is made of the master Thermodynamic Data tape and those species that are consistent with the chemical system under consideration are selected. As the thermodynamic data are being selected, the subroutine also complies a set of formula numbers, a_{ij}, from the formulas of the reaction products. A short Thermodynamic Data file is also generated for use in subsequent calculations.

A check is made near the beginning of the routine to prevent THERMØ data from exceeding their storage allotments. These variables are all in labeled common SPECIES and are currently dimensioned for 150 species. However. this dimension may be reduced to save storage.

SEARCH is called from the main program when the logical variable NEWR is true. NEWR is set true in REACT to indicate a new chemical system. REACT also stores chemical element symbols for the current chemical system in the LLMT array. SLARCH stores THERMØ data in core for each species whose elements are included in the LLMT array (unless the species name was listed on an ØMIT card).

The THERMØ data are stored in common variables TLØW, TMID, THIGH, SUB, A, CØEF, and TEMP. SEARCH writes out the names and dates of species whose data are stored in core.

SEARCH initializes the IUSE array. IUSE(j) for gaseous species are set equal to zero. IUSE(j) for condensed species are set equal to negative integers. For the chemical system under consideration the first possible condensed species is set equal to -1, the second to -2, and so on, with one exception. In the event there are two or more condensed phases of the same species, set equal to -4, for example, IUSE(j) for B_2O_3(s) will also be set equal to -4. A description of the IUSE array is given below.

The various condensed phases of a species are expected to be adjacent in the THERMØ data as they are read from tape. These phases must be either in increasing or decreasing order according to their temperature intervals.

NS contains the total number of species stored in core. NC contains the total number of condensed species (counting each condensed phase of a species as a separate species).

IUSE array. - Each value in the IUSE array is associated with a species. These values of IUSE serve two purposes:

1. They indicate which species are to be included in the current iteration (IUSE(j) < 0 for excluded species and IUSE(j) ≥ 0 for included species).
2. They indicate multiple phases of the same species if absolute values of IUSE(j) are equal.

The IUSE(j) are initialized in subroutine SEARCH and the main program as follows:

1. IUSE(j) = 0 for all gaseous species.
2. IUSE(j) = n for all condensed species whose names have been listed on INSERT cards. The number n indicates the species was the n^{th} condensed species whose THERMØ data were read from tape.
3. IUSE(j) = -n for all condensed species not listed on INSERT cards where n is defined in 2.

These initial values of IUSE(j) may be adjusted later in subroutine EQLBRM. For condensed species, the sign is adjusted as species are included or excluded in the current iteration.

For the IØNS option, IUSE(j) values for ionic species are set to -10000 when the mole fractions of all ionic species are less than 10^{-8}.

5.3.20 Subroutine SHCK

Subroutine SHCK is a dummy routine which replaces subroutine SHCK of Reference 1.

5.3.21 Subroutine THERMP

Subroutine THERMP is a dummy routine which replaces subroutine THERMP of Reference 1.

5.3.22 Subroutine VARFMT

Subroutine VARFMT (variable format) adjusts the number of decimal places printed in F-format in the variable format, FMT, according to the size of the number. It is used for P_c/P_e, P, and A_e/A_t. Variable format is described in BLØCK DATA, Section 5.1.2.

5.4 Link 30 Subroutines

The Link 30 subroutines control the grain design (GD) and ballistics (BAL) calculations made in the SPP code. These two modules have been integrated together from the programs of Reference 2 and Reference 6. The remainder of this subsection gives an overall description of the integrated ballistics module.

5.4.0.1 General

The Interior Ballistics Module models the internal ballistics of solid rocket motors, including motor environmental effects on burning rate (erosion, radiation), effects of radial slots between segments on the port flow, port flow pressure drop, and delayed burning (ignition) in deep narrow slots. Semi-empirical correlations for combustion inefficiency and nozzle throat erosion also are included. This computer module was adapted from an existing model used to analyze nozzleless solid rocket motors (Ref. 6), because it has gasdynamic features lacking in many published internal ballistics programs. The intent was to predict internal ballistics for motors ranging from small tactical to large boosters wherein aspects of these features are encountered. The major effort in the present program development involved updating certain model expressions, and rendering the computational procedure compatible with the Grain Design Module and the rest of the SPP code. The model will be subject to continued revision as new information becomes available.

The logic of solving the ballistics equations is presented in the following subsection. A discussion of the module subroutine structure is presented after that. Analogous discussion of the grain design subroutine structure is contained in Ref. (2) and need not be repeated here. Finally, the details of program input and output, for the integrated subprogram, are given in Volume III, The Program User's Manual.

The overall geometry-aerodynamics interface between the GD&BM is the generation of port area change as a function of length in the Grain Design routines and return of web burned from BM to GD module, at the same length stations. Ballistic calculations may be performed for one or more time increments within an increment of burn (NB) specified in the GD input, but as the increment is predicted to burn out the actual web expended is returned as the starting point for the next increment of geometry calculation.

5.4.0.2 Solution Logic

The general scheme employed in solving the equations of one-dimensional flow is shown in simplified terms in Figure 5.4-1. The procedure that follows the initialization of a case is as follows:

- A head-end pressure is assumed. The calculations for a single axial increment are made and iterated to completion. The successive increments down the port are then calculated until a match point on the nozzle entrance is reached.[1]
- At the match point, the local static pressure is checked against the pressure from throat flow constraints. If the two values are not within tolerance, a correction to head-end pressure is made and the calculation is restarted at the head. As successive iterations are made to satisfy the choke constraint (outer loop), the axial stations (inner loop) are continuously iterated.
- Parameters used in transient terms and integrating calculations are saved, and the time increment is advanced. The calculations resume with a new head-end pressure assumption.
- As the calculations reach the end of a specified run time, or the motor has burned out, output is furnished to the Nozzle Performance Module.

The computational logic, thus, is arranged as a triple-loop structure with the axial increment (inner loop), choke constraint (outer loop) and time increments (stepping).

A more detailed diagram of program logic is shown in Figure 5.4-2. This representation is still simplified with regard to the individual branching for special cases, but is intended to describe the function of groupings of program steps internal to the program, and presumes valid input. Figure 5.4-2 is organized in the same order as the source program.

The compiled-in constants are established and some core areas are cleared to zero.

A subroutine to control the reading and organizing of tabular data is called by Subroutine READIN on completion of geometry inputs. The remaining data (integers and real data input) are returned in intermediate storage which is inspected for non-zero (nz) values. Any nz values are then placed in working storage for use in MAIN 3.

---

[1] For an end-burner, the "head end" will be the grain face. A protective statement was included providing that if the motor port/throat is less than 1/4, the calculation will proceed to the next increment. Thus, the end face is eventually located. Internal burners are unlikely to have such low port/throat, so this provision should not introduce a problem in internal burners.

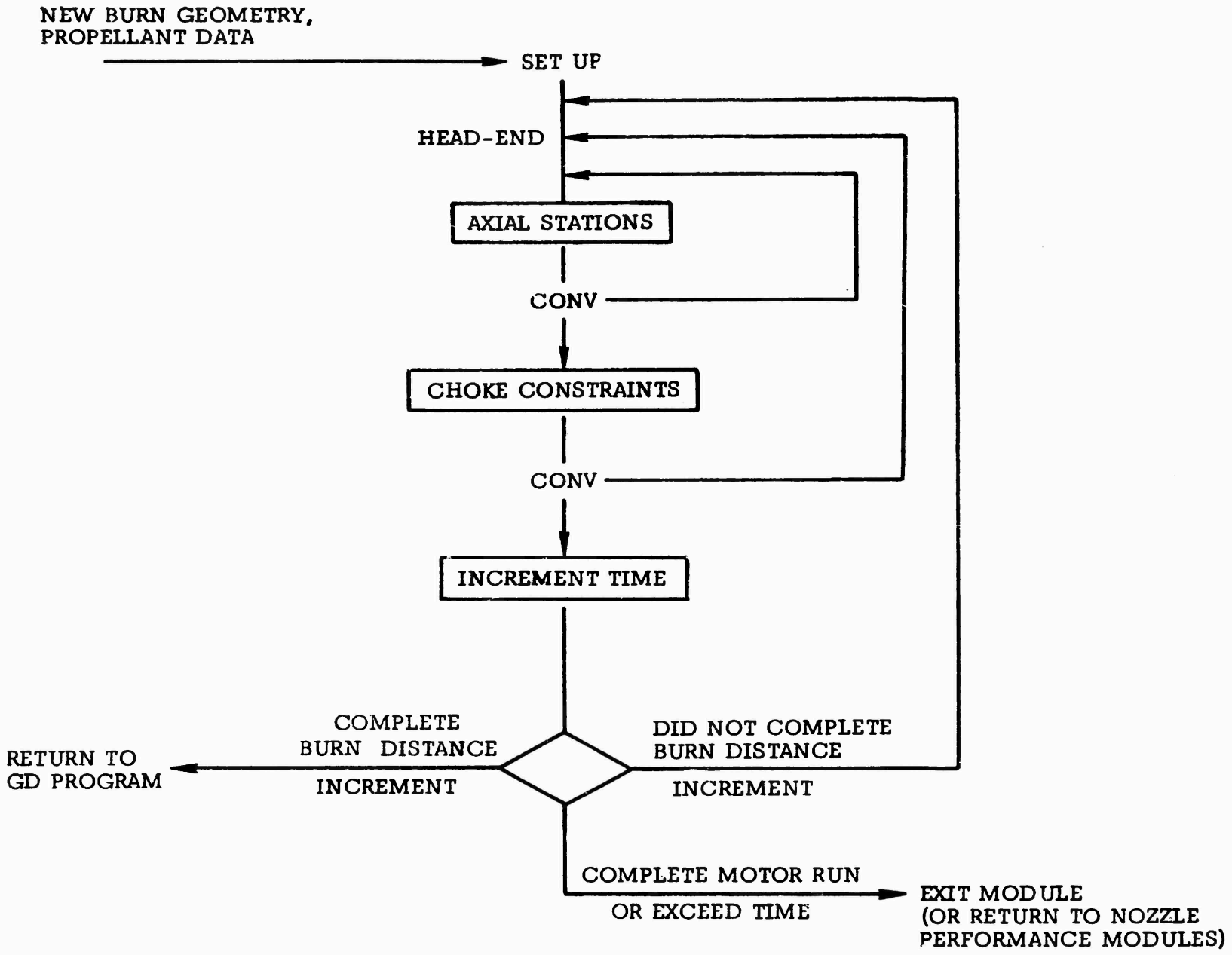


Figure 5.4-1 Simplified Logic

The outside diameter or periferal geometry of the grain is established from tabular inputs which are fed to subroutine LØØPS in the GD program. The GD program initializes the grain and computer burnback geometires.

The following groups of calculations are repeated at each time increment of the problem. The igniter flow, if any is determined, and time dependent values of A_t, Ae, thrust, ambient pressure and thrust coefficient are determined either from tabular input or internal feedback. Calculations are made for each axial increment in the motor. The axial increments correspond to the GD X-grid plus an aft plenum, plus the throat. Trial values for the initial time step are used from the previous axial station; all other trial values result from the previous iteration of the station. Special case logic for ignition options, or initial time are not shown in the figure or discussed here. (However, they should be recognized in the program listing as IGN.GT.0, or TIME.EQ.0. branches.)

The port geometry is calculated by the GD module from the web burned. Results are interpolated within the burn interval to determine local burn volumes.

- Base strand burn rate for the propellant at the axial station, as a function of local pressure, is found from an input table. The input data can be keyed to use log-log interpolation. The motor environment contributions are then calculated. π_K effects are incorporated by changing the base table.
- If appropriate, logic and branching related to radial slots is performed. The logic path returns several places depending on the interpretation of radial slot flow.
- Local conditions are calculated using the energy, state, and continuity equations, and thermochemical data.
- Traps are executed to prevent crossing Mach 1, and the momentum equation is used to solve for pressure. The set of equations at each axial station is iterated with logic for subsonic flow or Mach 1, with generation of an error term to later correct head pressure; excess iterations will be shown as output.
- Looping logic allows 10 iterations and then accepts the result whether converged or not.
- The node number of the axial increment is checked to branch to the next node or to throat calculations.
- Throat constraints are calculated from mass flow rate, nozzle throat area, C*, etc.
- Nozzle entrance static pressure is calculated from throat stagnation pressure, assuming an isentropic expansion, to compare with the pressure determined

by integration down the port. An updated head pressure is calculated from the error in nozzle inlet static pressure, or a term is created to indicate an overchoked prior calculation.

- If the throat constraint is not satisfied, loops are counted to 10. From 10 to 20 loops, the error is printed; after 20 loops, the result is accepted. The printed error messages indicate whether the calculations are oscillating or are overdamped and allow the user to determine whether the run is acceptable.

- Values used in transient terms or integrations are stored for use in the next time step. Parameters along the motor are printed. The burn-back of radial slots and web are updated. Axial nodes are coded out just prior to burn-back of the radial slot face to the node position. For radial slot combustion the slot ends are input to the GD program as non-burning. The burn-back is then explicitly calculated in MAIN3 and the axial increment locations are modified.

- Nozzle exit parameters and thrust are calculated from isentropic expansion equations, and a series of running integrals are calculated and output.

- The values of time and input run time are checked for completion. The next time increment or rewind follows this branch.

The indexing scheme employed in the program is shown in Table 5.4-1 as an aid in following the details of the logic. This logic is designed to treat non-perpendicular burn-back of radial slots; but is also used to "key" pressure drop calculations such as a sudden expansion or non-diffusing taper (i.e., separated flow).

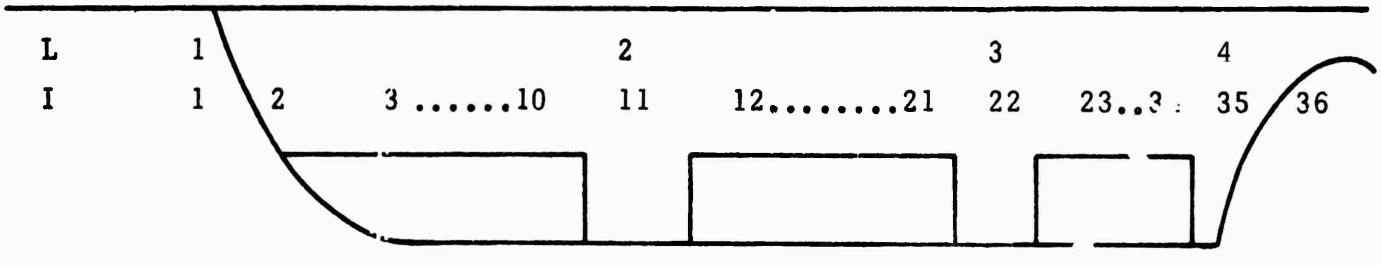


Table 5.4-1 INDEX SCHEME

Symbol	Description
I	Axial node location of calculation.
J	Prior axial node, usually J=I-1, J>0, but is varied to skip cast-off nodes (for example J=I-4 if three nodes are burned out).
L	Radial slot number, 1 at head end; slot relates to following segment.
N	Node number of nczzle entrance (36 in sketch).
NN	Throat node.

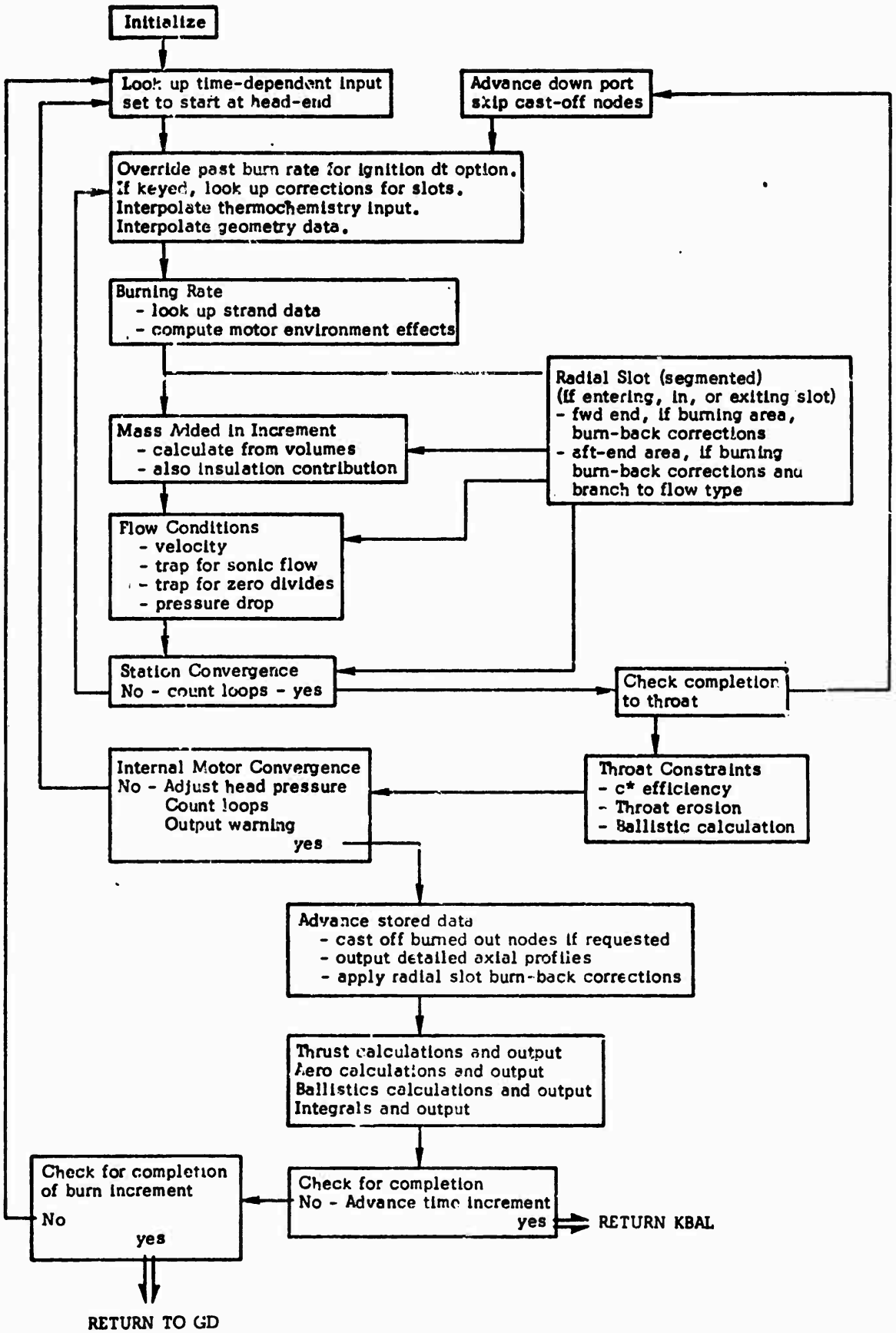


Figure 5.4-2 Module Structure

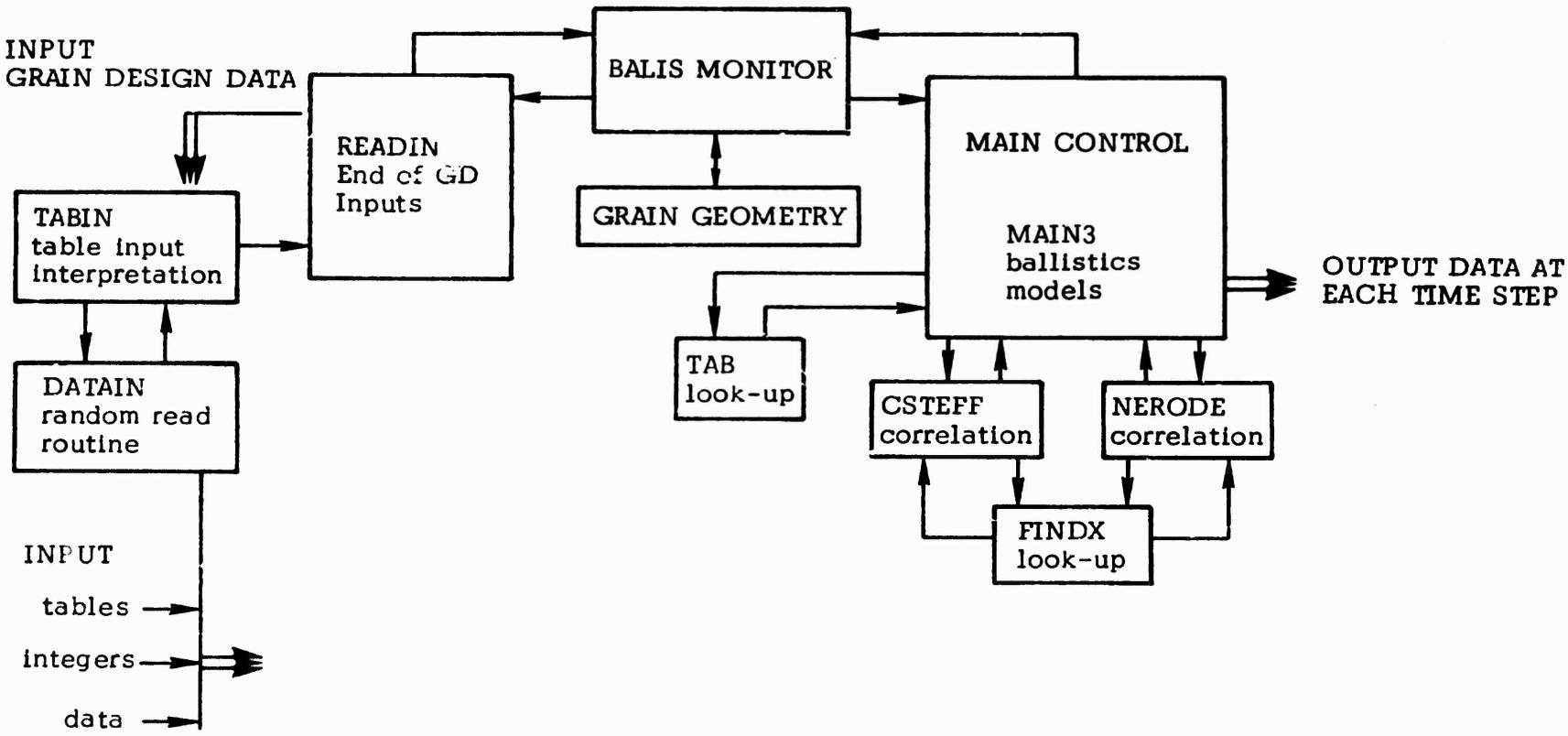


Figure 5.4-3 Module Logical Structure

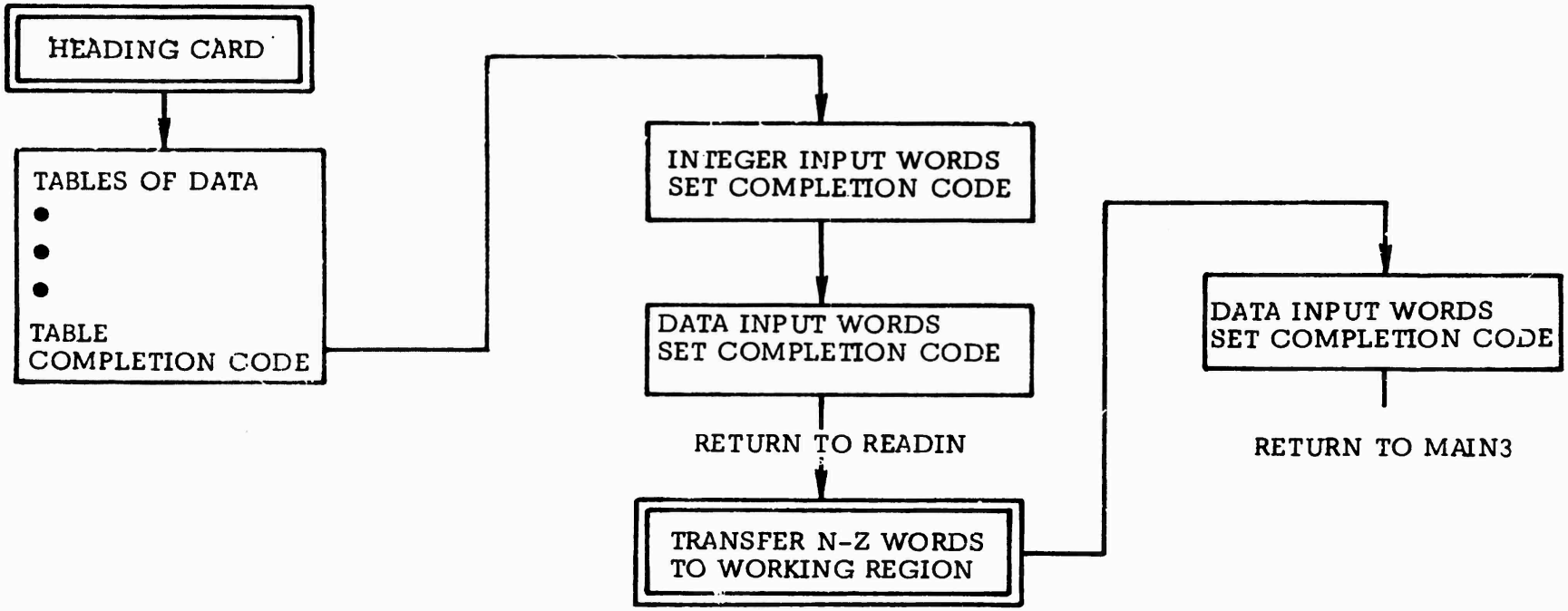


Figure 5.4-4 Input Groupings Following Grain Geometry Cards

5.4.1 Program LINK40

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to other computer systems. Its only function is to call subroutine BALIS.

5.4.2 Subroutine BALIS

This subroutine reads and writes the namelist data set $BAL. It then calls subroutine READIN and FIG1 to read in and then calculate the grain design parameters. Control of the ballistics and grain design calculations sequences are then handled in subroutines MAIN3, PVS and LØØPS. MAIN3 performs the axial looping for ballistics calculations, while LØØPS does the same for the grain geometry calculations.

5.4.3 BLØCK DATA

This routine is used to zero out the common blocks used in this module. The KALPHA array in labeled common CØDE is also initialized.

5.4.4 Subroutine CØNTR1

This subroutine has been written for use in the Grain Design Program as an auxiliary program for CONTR2. The routine computes the initial data needed by CONTR2 that is dependent only on the type of cone (or cylinder) being considered and independent of the burn distance, X plane, or line in question.

5.4.5 Subroutine CØNTR2

This subroutine has been written for use in the Grain Design Program. The purpose of the routine is to determine the points of intersection, if any, made by a line and a frustrum of a cone (or cylinder) located in three dimensional space.

5.4.6 Subroutine C3TEFF

This subroutine calculates the empirical c* efficiency described in Volume I, Table 4-2.

5.4.7 Subroutine DATAIN

DATAIN is a random read routine that interprets input of selected variables. DATAIN operates under the control of TABIN, which requests the reading of cards with integers or real data, which is in turn called from Subroutine READIN.

5.4.8 Subroutine ERROR

This subroutine was written for use with the Grain Design Program. The routine checks the input data for errors and writes error messages. See Appendix A for conversion aides.

5.4.9 Subroutine FIG1

Subroutine FIG1 has been written for use with the Grain Design Program. The program calls the figure routines for initial calculations. The initial burn used to input geometric figures with rounded corners is monitored by subroutine FIG1.

As Subroutine FIG1 is entered, I is initialized to one for the first figure. ID is computed from IDENT(I) and a "computed go to" is executed on ID in order to call the proper figure routine (CONTR1 or PRISM1). On return from the figure routine, I is incremented by the proper amount and a test of I against NOSRF is made to determine if more figures remain. Upon completion of the figures, control is returned to the main program.

5.4.10 Subroutine FIG2

Subroutine FIG2 has been written for use with the Grain Design Program. The program calls the proper figure routines for computing the points defining the void volume of the propellant.

As Subroutine FIG2 is entered, the motor case boundaries are computed for the specified Y value being considered. Then the burn distance to be considered (TRIAL) and the identification code (ID) for the figure to be considered are computed. Then the proper figure routine is called and upon return Subroutine UNION is called to form a union between the computed Z points. These points represent the shape of the propellant of the motor case.

If all input figures have not been covered, the loop is repeated, from the computing of TRIAL, for the next input figure.

5.4.11 Subroutine FINDX

This routine performs linear interpolation or extrapolation in tables while saving its place in the table of independent variables.

5.4.12 Subroutine LØØPS

Subroutine LØØPS sequences the grain design subroutines through the input lists of Y increments and performs a summation of port area contributions for the N:i: axial station in the motor. This subroutine replaces several in the original program of Ref. (2) since less options are retained for ballistics calculations and inverted order of sequencing is not needed. Thus, in the revised version, the outer loop is always burn increment; the middle loop is always motor axial station (dimension x); and the inner loop is on the radial intersection (dimension y). The subroutines performing the actual geometry calculation are essentially unchanged from Ref. (2); changes were introduced in the sequencing routines to imporve program run times.

5.4.13 Subroutine MAIN3

This routine controls the calculation of both the ballistics and grain design modules. In fact it is the main portion of the ballistics calculation.

Subroutine MAIN3 replaces a series of subroutines controlled by MAIN3 of the original Ref. (2) program. This subroutine calculates internal motor ballistics based on grain geometry output from the grain design subroutines. Control is retained in MAIN3 for as many time increments of web burned as necessary to predict burnout of the particular increment of web calculated by the grain design program. The web calculated by the ballistics subroutines is then returned to the grain design program through subroutine BALIS for use as the starting web in the next increment.

An artificial point in the nozzle entrance region is introduced as a match point for calculations proceeding down the port to the nozzle, for the purpose of testing the choking constraint without having to calculate flow near Mach one.

The area of the "nozzle entrance" is not critical to program operation, but run time is reduced if a large area is selected. The pressure computed by proceeding down the port must agree with that computed (by assuming isentropic flow) from the throat back to the artificial point, within a certain tolerance.

Values of c* efficiency and throat erosion, which are output from the ballistics subroutines, are based on semi-empirical correlations. Thrust may be calculated by the ballistics subroutines standing alone, but requires an input nozzle efficiency. This nozzle efficiency may be calculated external to the computer program, using such empirical expressions as contained herein or elsewhere. A series of motor operating parameters and running integrals are output and saved for use in other modules. If the other modules are exercised, the thrust parameters calculated herein are overridden.

Detailed output of conditions at each axial node in the calculations may be requested as a function of time using tabular input of the increments desired.

Calculation of the burn-back of radial slots is made by:

1) entering the slot faces as "nonburning" in the grain design program input;
2) overriding the grain inner diameter calculation with the outer diameter in the slot; and
3) explicitly calculating the location of the end of the port in subroutine MAIN3

The node at the end of the grain retains its identity number but is then a moving node. As the burning face approaches the next x-node along the grain, the interior node is coded out. Table input is provided to allow for ignition delay in the slot (radius ignited versus time). If significant delay is used, the face of the grain will not remain perpendicular to the motor axis (axial position is retained at the inner diameter); additional table input is then needed to correct the grain length as the port burns outward. Evidence that corrections are needed is that the total propellant weight expended will be under-predicted. The burning rate on the slot face is assumed to be the input strand rate.

5.4.14 Subroutine NERØDE

This routine calculates the nozzle throat erosion rate from input tables or from a correlation determined from analytical calculations.

5.4.15 Subroutine PRISM1

This routine calculates the initial data needed for each specific prism. Only right triangular prisms are considered.

5.4.16 Subroutine PRISM2

This subroutine has been written for use with the Grain Design Program. The purpose of the routine is to determine the two points of intersection, if any, made by a vertical line and a triangular prism located in three-dimensional space.

When a prism burns out, the corners burn into spheres, the edges burn into cylinders, and the end and side planes burn straight out. A line will intersect a prism either through the cylinder, sections (edges), sphere portions (corners), or not at all.

This routine considers the intersecting line as always a vertical line in the standard three-dimensional system. Hence, the vertical line is defined by an X and a Y vlaue. The number of line intersections necessary depends on the particular prism being considered and the accuracy desired in subsequent computations.

Input to this subroutine consists of an X value, a Y value, a T (burn) distance (positive for out-burning, negative for in-burning), and those computations made in PRISM1.

5.4.17 Subroutine PVS

This subroutine is used only with the ballistics mode. It provides peripheries, volumes, flow areas, and burning surfaces areas (and totals) at or between each X-station and burn interval ($time_k$ to $time_{k+1}$).

Cross sectional flow areas at $time_k$ and $time_{k+1}$ are supplied to this routine for each X-station. These are provided in two tables thus:

$A_{1,j}$ = Cross sectional flow area at $time_k$ and X_j.

$A_{2,j}$ = Cross sectional flow area at $time_{k+1}$ and X_j.

The n X-stations' locations are provided in a table of X_j.

Volume burned between each two adjacent X-stations during $time_k$ to $time_{k+1}$ is computed and summed:

$$V_{total} = \sum_{j=1}^{n-1} \Delta V_j$$

$$\Delta V_j = .5 \left[(A_{2,j} - A_{1,j}) + (A_{2,j+1} - A_{1,j+1}) \right] \cdot |X_{j+1} - X_j|$$

The average periphery of the flow area between $time_k$ and $time_{k+1}$, at X_j is calculated as the change in flow area over distance burned at that X-station. This is calculated only to be printed.

$$P_j = \frac{A_{2,j} - A_{1,j}}{\Delta B_j}$$

The average burning surface area during the interval $time_k$ to $time_{k+1}$, and X_j to X_{j+1} is calculated and summed. For this calculation, the equation $S=\Delta V/\Delta B$ had to be modified because ΔB is not necessarily normal to S in the ballistics mode.

5.4.18 Subroutine READIN

Subroutine READIN has been written for use with the Grain Design Program. The program initializes variables and reads the data. A test is made on the data in order to pass control to the appropriate place for storage of the data.

5.4.19 Subroutine SFERE2

This subroutine has been written for use with the Grain Design Program. The routine will determine the intersection of a line with a sphere located in three-dimensional space.

The routine solves the equation

$$C = R^2 - (Y_C - Y_1)^2 - (X_C - X_1)^2$$

where R = The radius of the sphere.

X_C = Denotes the location of a plane in three-dimensional space.

Y_C = Denotes a line in the X plane.

Y_1 = The Y coordinate describing the center of the sphere.

X_1 = The X coordinate describing the center of the sphere.

If C is negative, the line does not intersect with the sphere.

If C is zero, the line is tangent to the sphere.

If C is positive, the intersection points of the line with the sphere are given by:

$$Z = Z_1 + C$$

where Z_1 = The Z coordinate describing the center of the sphere.

5.4.20 Subroutine STORE

Subroutine STORE has been written for use with the Grain Design Program. Its purpose is to determine the nature of the input data and store it in locations to which the main program has access.

Control is sent to the program from Subroutine READIN by the statement CALL STORE (J,JO); JO has been assigned an integer value of 1 to 4 by Subroutine READIN depending on the card type and is used in a "computed go to". J is a subscript incremented in Subroutine READIN for the correct storage of data into an array.

5.4.21 Subroutine TAB

TAB is a routine to interpolate (or look up) values from the tabular inputs. Linear interpolation or log-log interpolation is used and no extrapolation is made; rather, the value at the boundary of the table is returned. The tables are assumed of the form

$$z = f(x,y)$$

with a call statement of the form

CALL TAB (X, Y, Z, No)

where No is a table number assigned in Subroutine MAIN3. The size, shape, type of interpolation, and order of table loading are established by input of the tables and thus are not transmitted to other subroutines.

Internally, the logic is established to search the table by stepping from the prior point to the new one, consistent with an iterative program with highly repetitious call statements, in contrast to computing indexes.

5.4.22 Subroutine TABIN

TABIN is a routine which prepares input of tabular data. The addressing for the tables is arranged, indexes are initialized, and limits to table size established. Input of integers and non-tabular program inputs are simply passed back to READIN for interpretation.

5.4.23 Subroutine UNIØN

This subroutine has been written for specific use with the Grain Design Program. The subroutine performs a union with Z points solved for previously and those Z points already entered in the table.

A table of void points or grain points is kept by entering, deleting, or in general, performaing a union with paris of Z points. These Z points represent the intersection of a line and the basic figures allowed for in the Grain Design Program. Based on a flag, the Z points can be treated either as cuts or fills to the table. Only those points equal to, or contained within, the case boundaries are considered. All points in the table are left adjusted. A count of the number of pairs is also computed.

Program Symbols

Fortran	Description
TABLE	Location where void or grain points are stored after union is made.
NOZE	Number or paris of Z in table.
RDIN(2)	Lower boundary or case limit points.
RDIN(3)	Upper boundary or case limit points.
Z(1) and Z(2)	The void or grain points.
ZEE and ZEET	Z(1) and Z(2).
ZF	Temporary stroage for one of void or grain points.
J	Number of points in table.
I	A count of number of points already looked at in table.
IUFA	Flag used to enter or delete Z points. This flag is also set outside of union to tell if table contains grains or void points.
KALI	A counter used to aid in entering or deleting points from the table.
IDIF	A difference value used to left-adjust points in the table.
KUFA	A branch flag used within the program.

5.5 Link 40 Subroutines

Overlay 40 contains the ØDK module which calculates the loss due to finite rate gas phase kinetics as incorporated into the methodology programmed in the SPP code.

This overlay includes several sub-overlays which are executed in sequence, i.e., Link 42 is executed after Link 41, etc. The subroutine descriptions included in this section are in alphabetical order within the individual overlay structure.

All of the subroutines described in this section are the results of the modification of the ØDK program described in Reference 3. While only certain of the subprograms of that reference have been modified, the entire pertinent body (i.e., ØDK documentation) of Reference 3 is repeated here.

5.5.1 Subroutine LINK40

This subroutine consists only of a call to subroutine ØDK and is used only to facilitate conversion of the program overlay structure to other computers.

5.5.2 Subroutine ØDK

This subroutine acts as the driver for the one dimensional kinetic expansion calculation (ØDK). It calls in overlays LINK41 through LINK43. Subroutine ØDK also writes (punches) out the linkage data dealing with the restricted equilibrium option of the ØDE module.

5.5.3 Subroutine STF

This subroutine evaluates the thermodynamic functions Cp°_T/R, H°_T/RT, S°_T/R, from curve fit coefficients. The subroutine uses the same procedure as subroutine CPHS. The additional functions $d(Cp^{\circ}_T)/dT$ and free energy, G°_T/RT, are also computed. The calculated functions are then converted to the internal computational units for use by the kinetic expansion calculations. Also, low temperature thermodynamic data from tables is used when required if the LTCPHS directive was specified in the input to the SPP code.

5.5.4 Subroutine STOICC

For each reaction this subroutine constructs two vectors of stoichiometric coefficients, one for reactants and one for products. Up to 10 reactants and 10 products may be considered for each reaction. The total number of entries in the resultant linear reaction table is 600, i.e., the sum of all stoichiometric coefficients cannot exceed 600.

5.5.5 Program Link 41

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to other computer systems. Its only function is to call subroutine ØDKINP which handles the input to the ØDK module.

5.5.6 Subroutine ECNV

This subroutine translates a BCD string of characters into one floating point numeric value. E,I, and F formats are permitted with the result always a floating point number. It is called by subroutine REAXIN to decode numeric fields in the species and reactions cards. The subroutine is coded entirely in FORTRAN. A BCD string of blanks will result in a floating point zero returned value.

5.5.7 Subroutine NUMBR

This subroutine converts a one character BCD number to a FORTRAN integer number. The subroutine is coded entirely in FØRTRAN.

5.5.8 Subroutine ØDKINP

This subroutine provides the input processing for the kinetic expansion calculation. It performs the following functions:

1) Variable initialization to nominal values

2) Calls subroutine REAXIN to input the reactions cards and species cards if necessary

3) For an ØDE-ØDK problem, calls subroutine SELECT to select those species to be considered for the kinetic expansion calculation

4) Reads $ØDK namelist input data

5) Converts nozzle geometric parameters from input units: inches, degrees; to internal computational units: feet, radians

6) Computes nozzle tangent coordinates using:

$$r_t = 1 + R_d (1 - \cos \theta)$$

$$x_t = R_d \sin \theta$$

7) For conical nozzles, computes the axial coordinate for the exit station from the following relation:

$$x_{exit} = \frac{\sqrt{\epsilon} - r_t + x_t \cdot \tan \theta}{\tan \theta} \qquad x_{exit} \geq x_t$$

$$x_{exit} = \left\{ R_d^2 - (1 + R_d - \sqrt{EP})^2 \right\}^{\frac{1}{2}} \qquad x_{exit} < x_t$$

8) For conical nozzles, the internal axial print stations are computed using:

$$X_j = \frac{\sqrt{ARPRNT(J)} - r_t + x_t \cdot \tan \theta}{\tan \theta} \qquad X_j \geq X_t$$

$$X_j = \left\{ R_d^2 - \left[1 + R_d - (ARPRNT(J))^{\frac{1}{2}} \right]^2 \right\}^{\frac{1}{2}} \qquad X_j < X_t$$

9) The sum of input or selected species concentrations is checked for unity ($\pm$ XMFTST, where XMFTST is an input number), and then normalized.

10) If the input parameter RZNØRM is input, the input contoured nozzle table is normalized by RZNØRM.

In addition to the above, the following modifications have been made to the ØDKINP routine.

a) If the nozzle geometry has been input via the $GEØM namelist, ØDKINP traps and stores the variables from this input into the correct locations and performs the appropriate unit conversions.

b) Was modified to accept the parabolic and circular arc options (IWALL=2 and 3) from the $GEØM namelist. This was done by computing the nozzle wall points and derivatives at 20 equally spaced points and substituting them into the tables used in the contoured wall input option.

c) Was modified to accept as input the quantities needed to calculate the zero particle lag flow equations.

5.5.9 Subroutine REAXIN

This subroutine processes SPECIES, REACTIØNS, and THIRD BØDY REAX RATE RATIØS input cards. Reference may be made to Volume III, Section 2.8.2, the Program User's Manual, for a complete description of input requirements. A table of all species appearing in the input reaction set is generated for further processing by subroutine SELECT if required.

5.5.10 Subroutine SELECT

This subroutine provides the interface logic required to select the minimum species list required for the kinetic expansion calculations. The subroutine is only used for the ØDE-ØDK interface. The list of all species appearing in the input reaction set is matched against the list of species considered for the equilibrium calculation. All species which appear in both a reaction and the equilibrium calculation list are selected for the kinetic expansion calculation. If a species appears in the reaction set but has not been considered for the equilibrium calculation, the program prints an error message and terminates the current case.

If the INERTS directive was specified those species specified under that directive will be added to the list for the kinetic expansion calculation along with the other species selected.

If the INERTS directive was not specified, all those species, considered for the equilibrium calculation, whose mole fractions are greater than or equal to an input selection criterion will also be selected for the kinetic expansion calculation. Species selected in this way will be listed as inert species on the program output since they do not enter into chemical reaction.

This routine was modified to select pairs of condensed phases if either of the phases passes one of the above selection criteria.

5.5.11 Program Link 4?

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to other computer systems. Its only function is to call subroutine PACK. See Section 5.5.13 below.

5.5.12 Subroutine CØNVRT

This subroutine converts input data from the externally input units to internally used computation units. In order to conserve computation time during the kinetic expansion, parameters such as molecular weights, are included in these conversions. Primed numbers are input quantities.

a) Reaction rate ratio input for reactions requiring third body terms
units: unitless
internal units: (lbs-mass/lb-mole)$^{-1}$
formula: $XMM_{j,i} = XMM_{j,i}/Mw_i$

b) Pre-exponential reaction rate parameter
input units: cm, °K, g-mole, sec
internal units: ft^3, °R, lb-mole, sec

$$a_j = \frac{a_j' \cdot (.0160183)^{\lambda} \cdot 1.8^{n_j}}{\prod_{i=1}^{n} Mw_i^{\nu_{ij}'}}$$

Where λ depends on the order of the reaction.

and $.0160183 = \frac{3.531 \cdot 10^{-5}\ ft^3}{1\ cm^3} \cdot \frac{1\ g\text{-mass}}{2.2 \cdot 10^{-3}\ lbs\text{-mass}}$

c) Exponential Term:
input units: kcal/mole
internal units: °R
formula: $b_j = b_j' \cdot 905.770$

where $905.770 = \frac{1000\ cal}{1\ kcal} \cdot \frac{1}{1.98726\ cal/mole\text{-}°K} \cdot \frac{1.8°R}{1.0°K}$

d) Equilibrium Constant Multiplicative Factor:

Input units: not input

Internal units: (lbs-mass) - $^{\circ}R/ft^3$

formula:

$$DATEF(j) = \frac{\prod_{i=1}^{n} Mw_i^{\nu_{ij}}}{\prod^{n} Mw_i^{\nu_{ij}} \cdot 0.73034}$$

where $.73034 = 49,721.011 \cdot \frac{\text{ft-poundals}}{(\text{lbs-mole})\ ^{\circ}R} \cdot \frac{1 \text{ atmos}}{68,059.59 \text{ poundals/ft}^2}$

e) Pressure:

Input units: PSIA

Internal units: $poundals/ft^2$

formula: $P = P' \cdot 4633.056$

where

$$4633.056 = \frac{144 \text{ in}^2}{1 \text{ ft}^2} \cdot 32.174 \frac{\text{ft}}{\text{sec}^2}$$

f) The initial reference enthalpy is computed using

$$H_{Ref} = \sum_{i=1}^{N} c_i h_i + \frac{V^2}{2}$$

Subroutine CØNVRT has been modified to calculate the total amount of condensed phase present at the kinetic expansion initial conditions.

5.5.13 Subroutine PACK

On the basis of those species currently being considered, this subroutine packs species and reaction information from the master tables into those control sections utilized by the one-dimensional kinetic expansion subprogram.

The following is a sequential description of the packing procedures:

1) Thermodynamic data for the species being considered is read into core storage.

2) The chemical species'molecular weights are computed

3) The symbolic reactions are checked for mass balance.

4) For a contoured nozzle the slope at each input wall point is computed using subroutine SLP. The wall coordinates, and each computed slope are printed for each input wall point and the print stations are set to the input axial coordinates.

In addition to the above, the PACK routine processes condensed phases by setting R_i=0 for those species and storing pointers up to 10 condensed phase species (5 pairs). This routine has also been modified to calculate the print positions for specified area ratios when the contoured (spline) wall option is selected.

5.5.14 Subroutine PRES

This subroutine is used (when JPFLAG = 1) to compute the derivatives of an input pressure table.

This subroutine is also used (when JPFLAG = 0) to generate a pressure table through use of an average expansion coefficient, Ne. The generated table extends from the initial contraction ratio through the nozzle attachment point plus one normalized throat radius.

Input Pressure Table Derivative Computation (JPFLAG = 1)

If a pressure table of NTB entries is input, the table of first derivatives is computed using:

$$\left.\frac{dP}{dx}\right|_{x_1} = 0$$

$$\left.\frac{dP}{dx}\right|_{x_n} = \frac{P_{(x_{n+1})} - P_{(x_{n-1})}}{x_{(n+1)} - x_{(n-1)}} \quad , \quad 1 < n < NTB$$

$$\left.\frac{dP}{dx}\right|_{x_{NTB}} = \frac{P_{(x_n)} - P_{(x_{n-1})}}{x_{(n)} - x_{(n-1)}} \quad , \quad n = NTB$$

The pressure at the initial axial position is obtained by interpolation using subroutine SPLN.

Internally Computed Pressure Table Computation (IPFLAG = 0)

An average equilibrium pressure expansion coefficient from the chamber to the throat, N_e, is computed by iteration using subroutine SUBNE. The initial value for $N_e^{(1)}$ is 1.2.

The approximate equilibrium contraction ratio at the initial axial position is computed from:

$$a_c = \left[\frac{N_e - 1}{2} \cdot \frac{\left[\frac{2}{N_e+1} \right]^{\frac{N_e+1}{N_e-1}}}{\left(\frac{P_i}{P_c} \right)^{\frac{2}{N_e}} \cdot \left[1 - \left(\frac{P_i}{P_c} \right)^{\frac{N_e-1}{N_e}} \right]} \right]^{\frac{1}{2}}$$

where P_i = pressure at the initial axial position

P_c = equilibrium chamber pressure

A check is then made to determine the compatibility between the nozzle geometry and the requested contraction ratio.

If

$$\sqrt{a_c} < 1 + \left[R_u + R_i \right] \cdot \left[1 - \cos \theta_i \right],$$

the circular arcs R_u and R_i overlap and the following error message is printed:

INLET GEOMETRY INCOMPATIBLE WITH INITIAL CONDITIONS

The program will proceed to the next case.

Tables for pressure and its derivatives are constructed as functions of area ratio, a, and expansion coefficient, Ne. Formula used for the j + 1 iteration for pressure is:

$$\frac{P^{(j+1)}}{P_c} = \frac{P^{(j)}}{P_c} + 2\left\{\frac{N_e-1}{N_e}\cdot\left[1-\left(\frac{P^{(j)}}{P_c}\right)^{\frac{N_e-1}{N_e}}\right]^{-1}\right.$$

$$\left.\left(\frac{P^{(j)}}{P_c}\right)^{-\frac{1}{N_e}} - \frac{2}{N_e}\cdot\left(\frac{P^{(j)}}{P_c}\right)^{-1}\right\}^{-1}$$

$$\cdot\left\{\left[\frac{N_e-1}{2}\cdot\frac{[2/(N_e+1)]^{(N_e+1)/(N_e-1)}}{\left(\frac{P^{(j)}}{P_c}\right)^{2/N_e}\cdot\left[1-\left(\frac{P^{(j)}}{P_c}\right)^{(N_e-1)/N_e}\right]}\right]^{-1/2}\cdot a^{-1}\right\}$$

for j = 1

$$\left.\frac{P^{(1)}}{P_c}\right|_{x_{n+1}} = \left.\frac{P}{P_c}\right|_{x_n} + \left.\frac{d(P/P_c)}{dx}\right|_{x_n}(x_{n+1}-x_n)$$

where n refers to the n^{th} table entry.

The pressure derivative formula used is:

$$\frac{d(P/P_c)}{dx} = \left[\frac{N_e-1}{N_e}\left(\frac{P}{P_c}\right)^{-\frac{1}{N_e}}\left[1-\left(\frac{P}{P_c}\right)^{\frac{N_e-1}{N_e}}\right]^{-1} - \frac{2}{N_e}\left(\frac{P}{P_c}\right)^{-1}\right]^{-1}\cdot\frac{2}{a}\frac{da}{dx}$$

Next, tables for pressure and its derivatives are constructed by the program. Table entries are at increments of

$-x_i/75$	for	$x_i < x < 0$
$(R_d \sin\theta)/25$	for	$0 < x < R_d \sin\theta$

and

$1/25$	for	$R_d \sin\theta < x < R_d \sin\theta + 1$

where the initial nozzle axial position, x_i, is computed from:

$$x_i = -\left[(R_u + R_i) \cdot \sin\theta_i + \frac{\sqrt{a_c} - 1 - (R_u + R_i) \cdot (1 - \cos\theta_i)}{\tan\theta_i}\right]$$

See Figure 5.5-1 below:

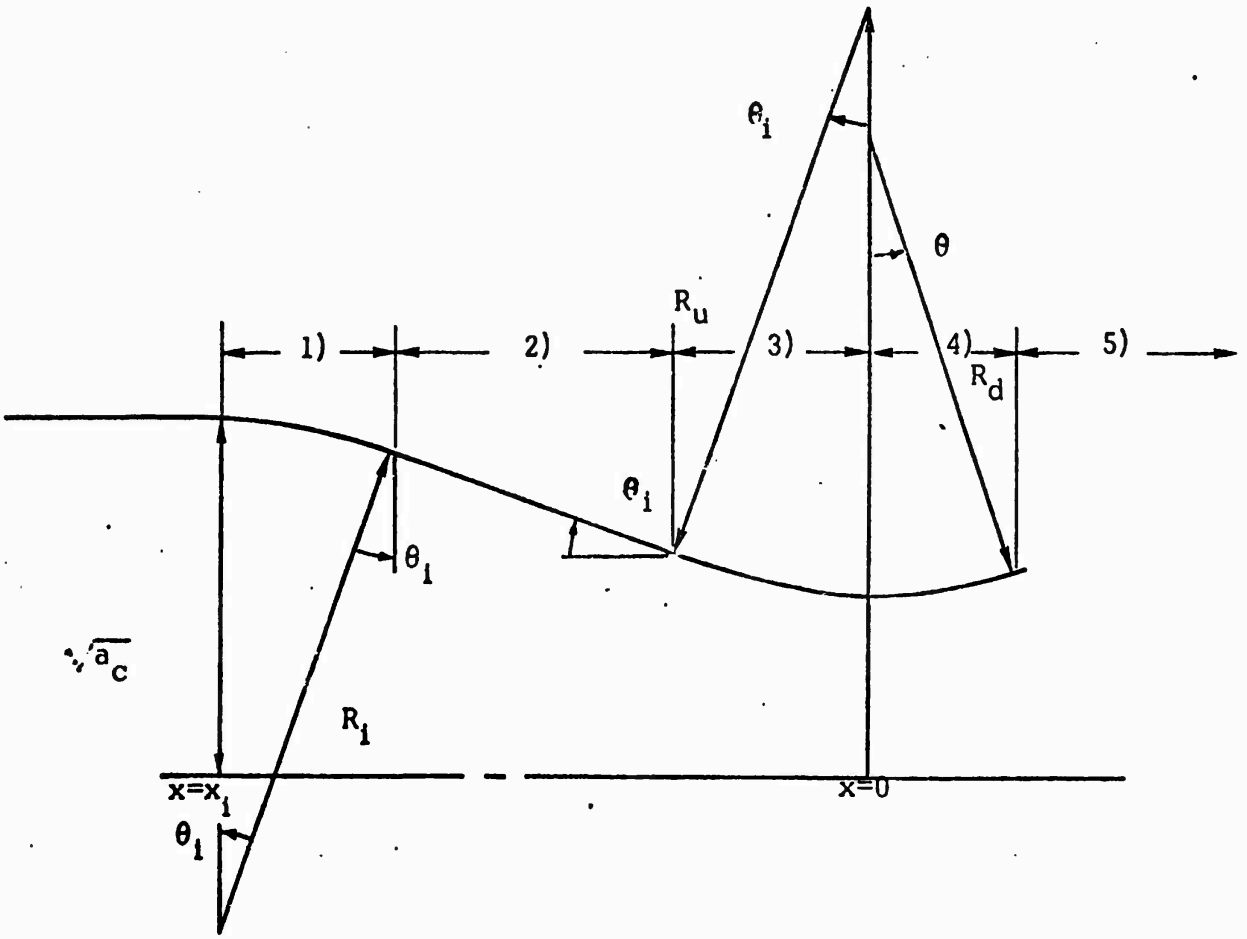


Figure 5.5-1 NOZZLE GEOMETRY

Area ratio and its derivative and (a and $\frac{da}{dx}$) are found by the five formulae below:

1) $x < x_i + R_i \sin\theta_i$

$$a = \left[\sqrt{a_c} - R_i\left(1 - \sqrt{1 - \frac{(x-x_i)^2}{R_i^2}}\right)\right]^2$$

$$\frac{da}{dx} = \frac{-2(x-x_i)}{\left[R_i^2 - (x-x_i)^2\right]^{1/2}} \cdot \sqrt{a}$$

2) $x_i + R_i \sin\theta_i < x < -R_u \sin\theta_i$

$$a = \left[\sqrt{a_c} - R_i(1 - \cos\theta_i) - (x - x_i - R_i \sin\theta_i)\tan\theta_i\right]^2$$

$$\frac{da}{dx} = -2 \cdot \sqrt{a} \cdot \tan\theta_i$$

3) $-R_u \cdot \sin\theta_i < x < 0$

$$a = \left[1 + R_u\left(1 - \sqrt{1 - \frac{x^2}{R_u^2}}\right)\right]^2$$

$$\frac{da}{dx} = \frac{2x}{\left[R_u^2 - x^2\right]^{1/2}} \cdot \sqrt{a}$$

4) $0 < x < R_d \cdot \sin\theta$

$$a = \left[1 + R_d\left(1 - \sqrt{1 - \frac{x^2}{R_d^2}}\right)\right]^2$$

$$\frac{da}{dx} = \frac{2x}{\left[R_d^2 - x^2\right]^{1/2}} \cdot \sqrt{a}$$

5) $R_d \cdot \sin\theta < x \leq R_d \cdot \sin\theta + 1$

for cone

$$a = \left[r_t + (x - x_t) \cdot \tan\theta \right]^2$$

$$\frac{da}{dx} = 2 \cdot a \cdot \tan\theta$$

for contour

$$a = Y^2$$

$$\frac{da}{dx} = 2 \cdot Y \cdot \frac{dY}{dx}$$

Three special points are included in the pressure table. These are a point at $x = x_i$ such that

$$\frac{P}{P_c} = \frac{P_i}{P_c}$$

$$\frac{d(P/P_c)}{dx} = 0$$

and two points at $x = 0$ such that

$$\frac{P}{P_c} = \left(\frac{P^*}{P_c} \right)_{\text{equilibrium}} \qquad \frac{3N_e - 1}{2(N_e - 1)}$$

$$\frac{d(P/P_c)}{dx} = - \frac{N_e}{\sqrt{R^*}} \cdot \left[\frac{2}{N_e + 1} \right]$$

with $R^* = R_u$ and $R^* = R_d$, respectively.

The following items are input directly to the computer program as described in Volume III, Section 2.8.5, Figure 2-2, left to right.

R_i	RI
R_u	RWTU
θ_i	THETAI
θ	THETA
R_d	RWTD

5.5.15 Subroutine SLP (X, Y, N, MFLAG, YP, W1, W2, W3, IFLAG)

The purpose of this subroutine is to supply derivatives for a tabulated function. The end point derivatives may be specified or are calculated internally by parabolic interpolation. Interior point derivatives may be found by a cubic spline fit procedure.

Calling Sequence:

X	is a table of independent variables, x_i
Y	is a table of the dependent variables, y_i
N	is the number of entries in each of the tables X, Y, and YP. $i = 1, \ldots N$
MFLAG	this entry is a flag, m, such that
	$m > 0$ implies x is equally spaced
	$m < 0$ implies x is not equally spaced
	$\|m\| = 1$ y' will be continuous
	$\|m\| = 2$ y' and y'' will be continuous
YP	is a table of the derivative, y'_i
W1	working storage of length N
W2	working storage of length N
W3	working storage of length N
IFLAG	this entry is a flag, i, such that
	$i = o$ implies value for YP(1) and YP(N) will be calculated internally by parabolic differencing
	$i = 1$ implies values for YP(1) and YP(N) will be input

Method

The cubic spline fit procedure utilizes the interpolation formula given below:

$$y = A(x - x_o)^3 + B(x - x_o)^2 + C(x - x_o) + D$$

$$y' = 3A(x - x_o)^2 + 2B(x - x_o) + C$$

$$y'' = 6A(x - x_o) + 2B$$

The piecewise cubic fit to a tabular function by the above relations will yield a discontinuity in the second derivative y'', between adjacent fits of:

$$y''_{1_{01}} - y''_{1_{12}} = \frac{1}{h_{01}}\left(2y'_0 + 4y'_1 - 6\,\frac{k_{01}}{h_{01}}\right) - \frac{1}{h_{12}}\left(6\,\frac{k_{12}}{h_{12}} - 4y'_1 - 2y'_2\right)$$

where

$$h_{01} = x_1 - x_0$$

$$h_{12} = x_2 - x_1$$

$$k_{01} = y_1 - y_0$$

$$k_{12} = y_2 - y_1$$

The method consists of setting the left-hand side of the above relation equal to zero so that the second derivative is continuous across juncture points. As applied to a tabular function, the above procedure results in a set of linear simultaneous equations (tri-diagonal) to be solved for the y'_i, provided that values for y' at the end points are known.

5.5.16 Subroutine SUBNE

Calculates the average equilibrium pressure expansion coefficient from the chamber to the throat by iteration from the following formula (Newton's method):

$$N_e^{(n+1)} = N_e^{(n)} + \frac{\left(\frac{2}{N_e^{(n)}+1}\right)^{\frac{N_e^{(n)}}{N_e^{(n)}-1}} - \frac{P_e^*}{P_c}}{\left(\frac{2}{N_e^{(n)}+1}\right)^{\frac{N_e^{(n)}}{N_e^{(n)}-1}} \left(\frac{1}{N_e^{(n)}-1}\right)\left[\frac{1}{N_e^{(n)}-1} \ln\left(\frac{2}{N_e^{(n)}+1}\right) + \frac{N_e^{(n)}}{N_e^{(n)}+1}\right]}$$

where $N_e^{(1)} = 1.2$.

P_e^* is the equilibrium throat pressure

P_c is the equilibrium chamber pressure

This subroutine is used by subroutine PRES.

5.5.17 Program Link 43

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to other computer systems. Its only function is to call subroutine MAIN1D which handles the control of the integration of equations describing one dimensional reacting gas flow through a rocket nozzle.

5.5.18 Subroutine EF

This subroutine computes equilibrium constants, K_j,

$$K_j \quad EK(J) = \frac{DATEF(J)}{T^{\lambda}} \cdot \exp\left[-\sum_{i=1}^{n} Ft_i \cdot \nu_{ij} + \sum_{i=1}^{n} Ft_i \cdot \nu'_{ij}\right]$$

also computed are

$$\frac{dK_j}{dT} \quad DKT(J) = \left[\frac{-\sum_{i=1}^{n} \frac{Ht_i}{R_i} \cdot \nu_{ij} + \sum_{i=1}^{n} \frac{Ht_i}{R_i} \cdot \nu'_{ij}}{T} - \lambda\right] \cdot \frac{K_j}{T}$$

where: Ft_i = species free energy at the current temperature

Ht_i = species enthalpy at the current temperature

DATEF(J) = is defined in subroutine CØNVRT, Section 5.5.12.

5.5.19 Subroutine DERIV

This subroutine computes the total derivatives f_i and the partial derivatives β_{ij} described in the analysis presented in Section 3 of Reference 3.

The implicit integration method used to integrate the differential equations governing the chemical system, i.e.

$$y_i' = f_i(x, y_1, \ldots, y_{NSP+3}) \qquad i=1, \ldots, NSP+3$$

where the variables

$$y_i \qquad i=1, \ldots, NSP+3$$

are $V, \rho, T, C_i \qquad i=1, \ldots, NSP$

respectively, requires evaluation of the Jacobian of the system, i. e.

$$\beta_{ij} = \frac{\partial f_i}{\partial y_j} \qquad \begin{matrix} i=1, \ldots, NSP+3 \\ j=1, \ldots, NSP+3 \end{matrix}$$

Subroutine DERIV computes only certain of the β_{ij} (those taken with respect to C_i) and the others are computed in subroutine FLU.

Also calculated by DERIV are the reaction rates, k_j, and the net production rates, X_j.

The generalized chemical reaction which is handled by this subroutine is defined by:

$$\sum_{i=1}^{NSP} \nu_{ij} \bar{M}_i \rightleftarrows \sum_{i=1}^{NSP} \nu'_{ij} \bar{M}_i$$

where $\bar{M}_i$ represents the i^{th} chemical species.

The reverse reaction rate constant is defined by the equation:

$$k_j \quad SK(J) \qquad = a_j \cdot T^{-n_j} \cdot \exp(-b_j/T)$$

The net production rate for a reaction is given by:

$$X_j \quad X(J) \qquad = \left[K_j \cdot \prod_{i=1}^{NSP} C_i^{\nu_{ij}} - \rho^{\lambda} \cdot \prod_{i=1}^{NSP} C_i^{\nu'_{ij}} \right] \cdot k_j \cdot M_j$$

where: λ depends on the order of the reaction*

with $M_j = \sum_{i=1}^{NSP} XMM_{j,i} \cdot C_i$ for reactions requiring a third body

and $M_j = 1$ for all other reactions

The net individual species production rate is given by the equation:

$$\frac{dC_i}{dx} \quad FN(I) \quad = \bar{K}_i \cdot \sum_{j=1}^{\ell} \psi_{ij} \cdot X_j$$

where:

$$\bar{K}_i = (Mw_i \cdot \rho \cdot r^*)/V$$

$$\psi_{ij} \equiv \nu'_{ij} - \nu_{ij}$$

The partial derivatives of the net species production rate with respect to: the chemical species; the gas velocity; the gas density; and the gas temperature are:

$$\beta(C_k, C_i) \quad BT(I,K) \quad = \quad \bar{K}_i \cdot \sum_{j=1}^{\ell} \frac{\partial X_j}{\partial C_i} \qquad \begin{matrix} i = 1, \ldots, NSP \\ k = 1, \ldots, NSP \end{matrix}$$

$$\beta(C_i, V) \quad PHI(I,1) \quad = \quad -\frac{1}{V} \frac{dC_i}{dx} \qquad i = 1, \ldots, NSP$$

$$\beta(C_i, \rho) \quad PHI(I,2) \quad = \quad \frac{1}{\rho} \cdot \frac{dC_i}{dx} + \bar{K}_i \sum_{j=1}^{m} \frac{\partial X_j}{\partial \rho} \qquad i = 1, \ldots, NSP$$

$$\beta(C_i, T) \quad PHI(I,3) \quad = \quad \bar{K}_i \sum_{j=1}^{\ell} \frac{\partial X_j}{\partial T} \qquad i = 1, \ldots, NSP$$

---

* $\lambda = \sum_{i=1}^{NSP} (\nu'_{ij} - \nu_{ij})$ so that $\lambda = 0$ for binary exchange, $\lambda = 1$ for most dissociation recombination reactions.

The subscript notation used above is:

i	=	Species subscript
j	=	Reaction subscript
ℓ	=	Total number of chemical reactions
m	=	Number of reactions requiring third body terms
NSP	=	Total number of gaseous species

Subroutine DERIV calls subroutine FLU to calculate the derivatives and partial derivatives of V, P, and T. In the event of solidification, this routine also recalculates the condensed phase species derivatives and their partial derivatives and recalls subroutine FLU to recalculate the flow derivatives. See Volume I, Section for the description of this procedure and the equations calculated.

5.5.20 Subroutine FLU

This subroutine computes the total derivatives f_i and the partial derivatives α_i and ε_{ij} for the fluid dynamic equations. While the flow is subsonic, pressure defined fluid dynamic equations are used. When the flow becomes supersonic, area defined fluid dynamic equations are used. The summation terms, energy exchange term B, the diabatic heat addition term A, the Mach number, and all the partial derivatives of these terms are computed. For a subsonic integration the pressure and its derivatives are obtained from the subsonic pressure table. For a supersonic integration, the area ratio and its derivatives are computed from the input geometric constraints.

The calculations logically fall into three types: a) Those done for all integrations; b) Those done only for subsonic integration; c) Those done only for supersonic integration. The following will adhere as closely as possible to a sequential description of the computations.

The operators $\Phi(i,l)$, $l = 1,2,3$ are defined as

$$\Phi(i,1) = \beta(C_i, V)$$

$$\Phi(i,2) = \beta(C_i, \rho)$$

$$\Phi(i,3) = \beta(C_i, T)$$

The total derivatives, $f_i = \frac{dC_i}{dx}$, for $i = 1, \ldots, n$ are computed as

$$f_i = \frac{\omega_i r^*}{\rho V}$$

where n is the total number of species, NSP.

Computation of the Summation Terms and their derivatives:

First Summation

S1	S1	$= \frac{1}{R} \cdot \sum_{i=1}^{n} \frac{dC_i}{dx} \cdot R_i$
$\frac{\partial S1}{\partial V}$	DS1V	$= \frac{1}{R} \cdot \sum_{i=1}^{n} \Phi_{(1,1)} \cdot R_i$
$\frac{\partial S1}{\partial \rho}$	DS1RØ	$= \frac{1}{R} \cdot \sum_{i=1}^{n} \Phi_{(1,2)} \cdot R_i$
$\frac{\partial S1}{\partial T}$	DS1T	$= \frac{1}{R} \cdot \sum_{i=1}^{n} \Phi_{(1,3)} \cdot R_i$
$\frac{\partial S1}{\partial C_l}$	DS1C(I)	$= \frac{1}{R} \cdot \left[\sum_{j=1}^{n} \beta_{(C_j, C_l)} \cdot R_j - S1 \cdot R_l \right] \quad l = 1, \ldots, n$

Second Summation

S2	S2	$= \frac{1}{R \cdot T} \cdot \sum_{i=1}^{n} \frac{dC_i}{dx} \cdot h_i$
$\frac{\partial S2}{\partial V}$	DS2V	$= \frac{1}{R \cdot T} \cdot \sum_{i=1}^{n} \Phi_{(1,1)} \cdot h_i$
$\frac{\partial S2}{\partial \rho}$	DS2RØ	$= \frac{1}{R \cdot T} \cdot \sum_{i=1}^{n} \Phi_{(1,2)} \cdot h_i$
$\frac{\partial S2}{\partial T}$	DS2T	$= \frac{1}{R \cdot T} \cdot \sum_{i=1}^{n} \left[\Phi_{(1,3)} \cdot h_i + \frac{dC_i}{dx} \cdot Cp_i \right] - \frac{S2}{T}$
$\frac{\partial S2}{\partial C_l}$	DS2C(I)	$= \frac{1}{R} \cdot \left[\sum_{j=1}^{n} \frac{\beta_{(C_j, C_l)} \cdot h_l}{T} - S2 \cdot R_l \right] \quad l = 1, \ldots, n$

*R_i is the gas constant/molecular wt. of species i except for condensed phases where $R_i = 0$.

Computation of the Energy Exchange Term B and its Derivatives:

$$B \qquad BB \qquad = \frac{\gamma-1}{\gamma} \cdot S2$$

$$\frac{\partial B}{\partial V} \qquad DBBV \qquad = \frac{\gamma-1}{\gamma} \cdot \frac{\partial S2}{\partial V}$$

$$\frac{\partial B}{\partial \rho} \qquad DBBR\emptyset \qquad = \frac{\gamma-1}{\gamma} \cdot \frac{\partial S2}{\partial \rho}$$

$$\frac{\partial B}{\partial T} \qquad DBBT \qquad = \frac{\gamma-1}{\gamma} \cdot \frac{\partial S2}{\partial T} + \frac{S2}{\gamma^2} \cdot \frac{\partial \gamma}{\partial T}$$

$$\frac{\partial B}{\partial C_i} \qquad DBBC(I) \qquad = \frac{\gamma-1}{\gamma} \cdot \frac{\partial S2}{\partial C_i} + \frac{S2}{\gamma^2} \cdot \frac{\partial \gamma}{\partial C_i} \qquad i = 1, \ldots, n$$

Computation of the Diabatic Heat Addition Term A and its Derivatives:

$$A \qquad AA \qquad = S1-B$$

$$\frac{\partial A}{\partial V} \qquad DAAV \qquad = \frac{\partial S1}{\partial V} - \frac{\partial B}{\partial V}$$

$$\frac{\partial A}{\partial \rho} \qquad DAAR\emptyset \qquad = \frac{\partial S1}{\partial \rho} - \frac{\partial B}{\partial \rho}$$

$$\frac{\partial A}{\partial T} \qquad DAAT \qquad = \frac{\partial S1}{\partial T} - \frac{\partial B}{\partial T}$$

$$\frac{\partial A}{\partial C_i} \qquad DAAC(I) \qquad = \frac{\partial S1}{\partial C_i} - \frac{\partial B}{\partial C_i} \qquad i = 1, \ldots, n$$

Computation of the Mach number and its derivatives:

$$M^2 \qquad XM2 \qquad = \frac{V^2}{\gamma \cdot R \cdot T}$$

$$\frac{\partial M^2}{\partial V} \qquad DM2V \qquad = \frac{2 \cdot M^2}{V}$$

$$\frac{\partial M^2}{\partial T} \qquad DM2T \qquad = -\frac{M^2}{T} - \frac{M^2}{\gamma} \cdot \frac{\partial \gamma}{\partial T}$$

$$\frac{\partial M^2}{\partial C_i} \qquad DM2C(I) \qquad = -M^2 \cdot \left[\frac{\partial \gamma}{\partial C_i} \cdot \frac{1}{\gamma} + \frac{R_i}{R} \right] \qquad i = 1, \ldots, n$$

For the subsonic portion of the nozzle, pressure defined fluid dynamic equations are used. The pressure, and its first and second derivatives are computed via interpolation in the pressure table generated by subroutine PRES. The Subsonic Gas Velocity derivatives are computed:

$$\frac{dV}{dx} \qquad FNX(1) \qquad = -\frac{1}{\rho \cdot V} \cdot \frac{dP}{dx}$$

$$\frac{\partial [FNX(1)]}{\partial x} \qquad AL(1) \qquad = -\frac{1}{\rho \cdot V} \cdot \frac{d^2P}{dx^2}$$

$$\beta(V,V) \qquad BETA(1,1) \qquad = -\frac{1}{V} \cdot \frac{dV}{dx}$$

$$\beta(V,\rho) \qquad BETA(1,2) \qquad = -\frac{1}{\rho} \cdot \frac{dV}{dx}$$

The Subsonic Gas Density derivatives are computed:

$$\frac{d\rho}{dx} \qquad FNX(2) \qquad = \rho \cdot \left[\frac{dP}{dx} \cdot \frac{1}{\gamma \cdot P} - \Lambda \right]$$

$$\frac{\partial [FNX(2)]}{\partial x} \qquad AL(2) \qquad = \frac{\rho}{\gamma \cdot P} \cdot \left[\frac{d^2P}{dx^2} - \left(\frac{dP}{dx} \right)^2 \cdot \frac{1}{P} \right]$$

$\beta(\rho,V)$ BETA(2,1) $= -\ \rho \cdot \frac{\partial A}{\partial V}$

$\beta(\rho,\rho)$ BETA(2,2) $= -\ \frac{1}{\rho} \cdot \frac{d\rho}{dx} - \rho \cdot \frac{\partial A}{\partial \rho}$

$\beta(\rho,T)$ BETA(2,3) $= -\ \rho \cdot \frac{\partial A}{\partial T} - \frac{\rho}{P \cdot \gamma^2} \cdot \frac{\partial \gamma}{\partial T} \cdot \frac{dP}{dx}$

$\beta(\rho,C_i)$ BETA(2,i+3) $= -\ \frac{\rho}{\gamma^2 P} \cdot \frac{\partial \gamma}{\partial C_i} \cdot \frac{dP}{dx} - \rho \cdot \frac{\partial A}{\partial C_i}$ $\quad i = 1,\ldots,n$

The Subsonic Gas Temperature derivatives are computed:

$\frac{dT}{dx}$ FNX(3) $= T \cdot \left[\frac{\gamma-1}{\gamma} \cdot \frac{1}{P} \cdot \frac{dP}{dx} - B\right]$

$\frac{\partial[\text{FNX}(3)]}{\partial x}$ AL(3) $= \frac{\gamma-1}{\gamma} \cdot \frac{T}{P} \cdot \left[\frac{d^2P}{dx^2} - \left(\frac{dP}{dx}\right)^2 \cdot \frac{1}{P}\right]$

$\beta(T,V)$ BETA(3,1) $= -\ T \cdot \frac{\partial B}{\partial V}$

$\beta(T,\rho)$ BETA(3,2) $= -\ T \cdot \frac{\partial B}{\partial \rho}$

$\beta(T,T)$ BETA(3,3) $= \frac{1}{T} \cdot \frac{dT}{dx} + T \frac{1}{\gamma^2 \cdot P} \frac{dP}{dx} \frac{\partial \gamma}{\partial T} - T \frac{\partial B}{\partial T}$

$\beta(T,C_i)$ BETA(3,i+3) $= T \cdot \left[\frac{1}{\gamma^2 \cdot P} \cdot \frac{dP}{dx} \cdot \frac{\partial \gamma}{\partial C_i} - \frac{\partial B}{\partial C_i}\right]$ $\quad i = 1,\ldots n$

For the supersonic portion of the nozzle, area defined fluid dynamic equations are used. The area ratio, and its derivatives are computed according to the input geometric constraints.

Area ratio and its derivatives:

1) On the circular arc of radius R_d (input item RWTD) defining the downstream throat region, $X \leq X_{tangent}$

$$a = \left[1 + R_d - \left(R_d^2 - x^2\right)^{1/2}\right]^2$$

$$\frac{da}{dx} = \frac{2x}{\left(R_d^2 - x^2\right)^{1/2}} \cdot \left[1 + R_d - \left(R_d^2 - x^2\right)^{1/2}\right]$$

$$\frac{d^2a}{dx^2} = \left[\frac{2}{\left(R_d^2 - x^2\right)^{1/2}} + \frac{2x^2}{\left(R_d - x^2\right)^{3/2}}\right] \cdot \left[1 + R_d - \left(R_d - x^2\right)^{1/2}\right] + \frac{2x^2}{R_d^2 - x^2}$$

2) For a conical nozzle and $X > X_{tangent}$

$$a = \left[r_t + \left(x - x_t\right) \tan \Theta_t\right]^2$$

$$\frac{da}{dx} = 2\left[r_t + \left(x - x_t\right) \tan \Theta_t\right] \cdot \tan \Theta_t$$

$$\frac{d^2a}{dx^2} = 2 \tan^2 \Theta_t$$

3) For contoured, parabolic, and circular arc nozzle and $X > X_{tangent}$

$$a = Y^2$$

$$\frac{da}{dx} = 2 \cdot Y \cdot \frac{dY}{dx}$$

$$\frac{d^2a}{dx^2} = 2 \cdot \left[Y \frac{d^2Y}{dx^2} + \left(\frac{dY}{dx} \right)^2 \right]$$

where Y, dY/dx are computed via interpolation in the tables of Y and its derivatives generated by exact calculations or by subroutine SLP.

The Supersonic Gas Velocity derivatives are computed:

$$\frac{dV}{dx} \qquad FNX(1) = \frac{V}{M^2-1} \cdot \left[\frac{1}{a} \frac{da}{dx} - A \right]$$

$$\frac{\partial[FNX(1)]}{\partial x} \quad AL(1) = \frac{V}{M^2-1} \cdot \frac{1}{a} \cdot \left[\frac{d^2a}{dx^2} - \frac{1}{a} \cdot \left(\frac{da}{dx} \right)^2 \right]$$

$$\beta(V,V) \quad BETA(1,1) = \frac{1}{V} \cdot \frac{dV}{dx} - \frac{1}{M^2-1} \cdot \frac{dV}{dx} \cdot \frac{\partial M^2}{\partial V} - \frac{V}{M^2-1} \cdot \frac{\partial A}{\partial V}$$

$$\beta(V,\rho) \quad BETA(1,2) = - \frac{V}{M^2-1} \cdot \frac{\partial A}{\partial \rho}$$

$$\beta(V,T) \quad BETA(1,3) = - \frac{1}{M^2-1} \cdot \frac{dV}{dx} \cdot \frac{\partial M^2}{\partial T} - \frac{V}{M^2-1} \cdot \frac{\partial A}{\partial T}$$

$$\beta(V,C_i) \quad BETA(1,i+3) = - \frac{1}{M^2-1} \cdot \frac{dV}{dx} \cdot \frac{\partial M^2}{\partial C_i} - \frac{V}{M^2-1} \cdot \frac{\partial A}{\partial C_i} \quad i=1,\ldots,n$$

The Supersonic Gas Density derivatives are computed:

$$\frac{d\rho}{dx} \quad \text{FNX(2)} \qquad = -\rho\cdot\left[\frac{M^2}{M^2-1}\cdot\left(\frac{1}{a}\cdot\frac{da}{dx}-A\right)+A\right]$$

$$\frac{\partial[\text{FNX(2)}]}{\partial x} \quad \text{AL(2)} \qquad = -\rho\cdot\frac{M^2}{M^2-1}\cdot\frac{1}{a}\cdot\left[\frac{d^2a}{dx^2}-\frac{1}{a}\left(\frac{da}{dx}\right)^2\right]$$

$$\beta(\rho,V) \quad \text{BETA(2,1)} = \rho\cdot\left[\frac{1}{(M^2-1)^2}\cdot\left(\frac{1}{a}\frac{da}{dx}-A\right)\cdot\frac{\partial M^2}{\partial V}+\frac{1}{M^2-1}\cdot\frac{\partial A}{\partial V}\right]$$

$$\beta(\rho,\rho) \quad \text{BETA(2,2)} = \frac{1}{\rho}\cdot\frac{d\rho}{dx}+\frac{\rho}{M^2-1}\cdot\frac{\partial A}{\partial\rho}$$

$$\beta(\rho,T) \quad \text{BETA(2,3)} = \rho\cdot\left[\frac{1}{(M^2-1)^2}\cdot\left(\frac{1}{a}\frac{da}{dx}-A\right)\cdot\frac{\partial M^2}{\partial T}+\frac{1}{M^2-1}\cdot\frac{\partial A}{\partial T}\right]$$

$$\beta(\rho,C_i) \quad \text{BETA(2,i+3)} = \rho\cdot\left[\frac{1}{(M^2-1)^2}\cdot\left(\frac{1}{a}\frac{da}{dx}-A\right)\frac{\partial M^2}{\partial C_i}+\frac{1}{M^2-1}\cdot\frac{\partial A}{\partial C_i}\right]$$

$$i = 1,\ldots,n$$

The Supersonic Gas Temperature derivatives are computed:

$$\frac{dT}{dx} \quad \text{FNX(3)} \qquad = -T\cdot\left[(\gamma-1)\cdot\frac{M^2}{M^2-1}\cdot\left(\frac{1}{a}\frac{da}{dx}-A\right)+B\right]$$

$$\frac{\partial[\text{FNX(3)}]}{\partial x} \quad \text{AL(3)} \qquad = -T\cdot\frac{M^2}{M^2-1}\cdot\frac{\gamma-1}{a}\cdot\left[\frac{d^2a}{dx^2}-\frac{1}{a}\cdot\left(\frac{da}{dx}\right)^2\right]$$

$$\beta(T,V) \quad \text{BETA(3,1)} = T\cdot\left[\frac{\gamma-1}{(M^2-1)^2}\left(\frac{1}{a}\frac{da}{dx}-A\right)\cdot\frac{\partial M^2}{\partial V}+\gamma-1\cdot\frac{M^2}{M^2-1}\cdot\frac{\partial A}{\partial V}-\frac{\partial B}{\partial V}\right]$$

$$\beta(T,\rho) \quad \text{BETA(3,2)} = T\cdot\left[\gamma-1\cdot\frac{M^2}{M^2-1}\cdot\frac{\partial A}{\partial\rho}-\frac{\partial B}{\partial\rho}\right]$$

$\beta(T,T)$ BETA(3,3) $= \frac{1}{T}\cdot\frac{dT}{dx} + T\cdot\left[\frac{\gamma-1}{(M^2-1)^2}\left(\frac{1}{a}\frac{da}{dx}-A\right)\frac{\partial M^2}{\partial T}\right.$

$+\gamma-1\cdot\frac{M^2}{M^2-1}\cdot\frac{\partial A}{\partial T} - \frac{\partial B}{\partial T}$

$\left. - \frac{M^2}{(M^2-1)}\cdot\left(\frac{1}{a}\frac{da}{dx}-A\right)\frac{\partial\gamma}{\partial T}\right]$

$\beta(T,C_1)$ BETA (3,1+3)= $T\cdot\left[\frac{\gamma-1}{(M^2-1)^2}\cdot\left(\frac{1}{a}\frac{da}{dx}-A\right)\frac{\partial M^2}{\partial C_1} + \gamma-1\cdot\frac{M^2}{M^2-1}\cdot\frac{\partial A}{\partial C_1}\right.$

$\left. -\frac{\partial B}{\partial C_1} - \frac{M^2}{M^2-1}\cdot\left(\frac{1}{a}\frac{da}{dx}-A\right)\frac{\partial\gamma}{\partial C_1}\right]$

$i = 1, \ldots, n$

5.5.21 Subroutine GTF

This subroutine computes the effective gas constant, gaseous heat capacity, γ, $\partial\gamma/\partial T$, $\partial\gamma/\partial C_i$ from the following formulae:

$$R = \sum_{i=1}^{NSP} C_i \cdot R_i$$

$$Cp = \sum_{i=1}^{NSP} C_i \cdot Cp_i$$

$$\gamma = \frac{Cp}{Cp-R}$$

$$\frac{\partial\gamma}{\partial T} = -\frac{\gamma\cdot(\gamma-1)}{Cp} \cdot \sum_{i=1}^{NSP} C_i \cdot \frac{\partial Cp_i}{\partial T}$$

$$\frac{\partial\gamma}{\partial C_i} = \gamma\cdot(\gamma-1)\cdot\left[\frac{R_i}{R} - \frac{Cp_i}{Cp}\right] \qquad i = 1, \ldots, n$$

For condensed phases, $R_i=0$, to account for the assumption that the particles exert no pressure on the gas.

5.5.22 Subroutine IAUX (HL, H, QK, RK, JX)

This subroutine performs implicit integration according to the method discussed in Section of Reference 3. The increments for the chemical species concentrations and the fluid dynamic variables at the forward point are calculated by solving the appropriate implicit finite difference formulas. Subroutine IAUX also performs explicit integration, using a modified Euler method, when the gas temperature falls below an input value.

The calling sequence parameters are:

HL - last integration step size

H - current integration step size

QK - last increments for variables

RK - computed increments for variables

JK - 1 initial 3 steps
2 general step
3 special step
4 restart step

The total derivatives, $f_{i,n}$, and partial derivatives, $\beta_{i,j,n}$ at the back point are calculated in subroutines DERIV and FLU.

The special step calculation is used at print stations, in halving the step size if required, or for integrating to specific calculation stations. If the special step calculation is used to determine the properties at a print station, the calculation is resumed using the general step calculation and the previous step size.

After each integration step, subroutine IAUX obtains the derivatives at the then current axial position.

For implicit integration the equations used are:

Initial Step and Restart

$$k_{i,1} = \left[f_{i,o} + \alpha_{i,o} h + \sum_{j=1}^{N} \beta_{i,j,o} k_{j,1} \right] \cdot h$$

General Step

$$k_{i,n+1} = \frac{1}{3} \left[k_{i,n} + 2 \cdot \left(f_{i,n} + \alpha_{i,n} h + \sum_{j=1}^{N} \beta_{i,j,n} k_{j,n+1} \right) \cdot h \right]$$

Special Step

$$k_{i,n+1} = \frac{h_{n+1}^2}{(2h_{n+1}+h_n) \cdot h_n} \left[k_{i,n} + \left[f_{i,n} + \alpha_{i,n} h_{n+1} + \sum_{j=1}^{N} \beta_{i,j,n} k_{j,n+1} \right] \cdot \frac{h_n}{h_{n+1}} (h_{n+1}+h_n) \right]$$

For explicit integration the above equations are used deleting the partial derivative terms α and β.

If the option to generate input tables for the Turbulent Boundary Layer Nozzle Analysis Computer Program was selected, tables of M, P/P_c, T/T_c, C_p, V, ρ, are tabulated using subroutine TABGEN. While the code remains in this routine to do the above, the SPP code does not use these values in any way.

5.5.23 Subroutine INT

Provides control for the implicit integration procedure, determines the proper set of nonhomogeneous equations to solve, and, after each integration step, computes the next integration step size according to the following relations:

$$h_{n+2} = 2h_{n+1}, \qquad \left| \frac{k_{i,n+1} - 2k_{i,n} + k_{i,n-1}}{3k_{i,n+1} - k_{i,n}} \right|_{MAX} < \frac{\delta}{10}$$

$$h_{n+2} = \frac{1}{2} h_{n+1}, \qquad \left| \frac{k_{i,n+1} - 2k_{i,n} + k_{i,n-1}}{3k_{i,n+1} - k_{i,n}} \right|_{MAX} > \delta$$

$$h_{n+2} = h_{n+1}, \qquad \frac{\delta}{10} \leq \left| \frac{k_{i,n+1} - 2k_{i,n} + k_{i,n-1}}{3k_{i,n+1} - k_{i,n}} \right|_{MAX} \leq \delta$$

On option, (JF=1) only the fluid dynamic variables are used in determining the next integration step size.

If the step size is halved for the fourth step, the integration is restarted using one-half the original step size.

The correspondence between equation number and physical property is:

Equation Number	Property
1	Velocity of Gas
2	Density of Gas
3	Temperature of Gas
4 → NSP+3	Gaseous species mass fraction
	(1 → NSP) corresponds to (4 → NSP + 3)

When the flow is supersonic, continuity is used to control the integration step size to insure that:

$$\left| \frac{(\rho VA)_{N+1} - (\rho VA)_N}{(\rho VA)_{N+1}} \right| < CØNDEL$$

where CØNDEL is an input relative criterion.

5.5.24 Subroutine LESK (Y)

This subroutine is a single precision linear equation solver which is used to perform the matrix inversions required by subroutine IAUX. Gaussian elimination is used with row interchange taking place to position maximum pivot elements after the rows are initially scaled.

5.5.25 Subroutine MAIN1D

This subroutine provides the overall logic control for the one-dimensional kinetic expansion. The following functions are controlled:

1) Variable initialization
2) Option to start the kinetic expansion from equilibrium throat conditions
3) Controls of the integration to hit specific area ratios, the nozzle throat point, the nozzle tangent point, and the requested end point
4) Controls of the switch from the subsonic pressure defined equations to the supersonic area defined equations when ($M^2 \geq 1.02$)
5) Controls the switch from implicit to explicit integration.

For the normal mode of operation of the program, this subroutine locates the throat in the following manner:

The gaseous mass flow per unit area (ρv) is calculated and stored as a function of nozzle axial location for the present and past integration step. When

$$(\rho v)_{n+1} < (\rho v)_n$$

where n refers to the n^{th} integration step, the throat location is calculated from:

$$X^* = X_n + \frac{(X_n - X_{n-1})^2 \cdot \left[(\rho v)_{n+1} - (\rho v)_n\right] + (X_{n+1} - X_n)^2 \cdot \left[(\rho v)_n - (\rho v)_{n-1}\right]}{2 \cdot \left[(X_{n+1} - X_n) \cdot \left[(\rho v)_n - (\rho v)_{n-1}\right] - (X_n - X_{n-1}) \cdot \left[(\rho v)_{n+1} - (\rho v)_n\right]\right]}$$

and the $n+1^{th}$ integration step is repeated using a step size of $X^* - X_n$ to determine the throat conditions.

To prevent the location of a false throat due to roughness of an input pressure table, ten integration steps are required before the throat will be sought.

Through the downstream throat radius of curvature the step size is controlled so as to be less than or equal to RWTD*SIN(THETA)/25.0,

Subroutine MAIN1D also contains the logic to control the integration through solidification of multiple condensed liquid phases if they are present. This logic allows the integration procedure to hit the beginning and ending of solidification exactly and to turn on the solidification equations in subroutine DERIV via the flag IFMELT in COMMON LKMELT.

5.5.26 Subroutine OUTPUT

This subroutine provides conversion from internal computational units to output engineering units and the calculation of performance parameters. The following output parameters are computed by this subroutine:

The pressure (in PSIA) is computed from:

$$P_{(PSIA)} = P/4633.056$$

The gaseous species mole fractions are computed from:

$$C_{i,m} = \frac{R_i}{R} \cdot C_i$$

The gas molecular weight is computed from:

$$Mw = 49721.011/R$$

The percentage mass fraction change is computed from:

$$\% \Delta (\text{Mass Fraction}) = 100.0 \cdot (1.0 - \sum_{i=1}^{n} C_i)$$

The gas heat capacity is computed from:

$$Cp_g(BTU/LB-^oR) = 3.9969 \cdot 10^{-5} \cdot Cp_g$$

The gas enthalpy is computed from:

$$H_g(BTU/LB) = 3.9969 \cdot 10^{-5} \cdot \sum_{i=1}^{n} C_i \cdot h_i$$

At the throat, the characteristic exhaust velocity (ft/sec) is computed from:

$$C^* = \frac{P_c}{\rho^* \cdot V^*}$$

The vacuum specific impulse is computed from:

$$ISP_{VAC} = \frac{V + P/P_c \epsilon C^*}{g} \quad g = 32.174$$

The vacuum thrust coefficient is computed from:

$$C_{F_{VAC}} = \frac{I_{sp_{vac}}}{C^*}$$

The percentage enthalpy change is computed from:

$$\% \Delta H_T = \frac{100 \cdot (HREF_c - HREF)}{V^2/2}$$

where

$$HREF = \sum_{i=1}^{NSP} C_i \cdot h_i + V^2/2$$

$HREF_c$ is HREF evaluated at the initial condition for the ØDK integration (i.e. at the initial contraction ratio, ECRAT).

In addition the actual molecular weight of the mixture of gases and condensed phases is also printed out.

5.5.27 Subroutine PRNTCK

For the option to print starting at step ND1, printing every ND3rd step up to step ND2, this subroutine checks whether or not the current step should be printed. If it is to be printed this subroutine calls subroutine ØUTPUT.

5.5.28 Subroutine TABGEN (IFLAG, LTABLE, XTAB, YTAB, LUSED, X, Y, IERRØR, NY)

This subroutine records a tabular function (X,Y(NY)) in tables of fixed length. The first and last event will always be tabulated and the table will either contain all of the values specified or will be at least half full. Once the number of events exceed the table length, the table will be repacked by the deletion of every other table entry and tabulation will proceed choosing every $2^{\underline{N}th}$ event(N=0, 1, 2. . . etc., where N is the number of times the table must be repacked). The table spacing will be a power of two except for the last event which will always be tabulated.

The calling sequence parameters are:

IFLAG	-	denotes type of entry to subroutine
		= - 1 first entry
		= 0 normal entry
		= + 1 last entry
LTABLE	-	length of tables available for tabulation
XTAB	-	table for tabulation of the variable X
YTAB	-	table for tabulation of the variable
LUSED	-	number of table entries currently used (output)
X	-	the variable X
Y	-	the variable Y
IERRØR	-	error flag
NY	-	number of Y variables to be tabulated

Note: One Dimensional Mach Number Tabulation Procedure

At the initial axial position, X and Mach number are recorded. TABGEN is then used with LTABLE=50 (assuring 25 saved values). The last recorded values are the end values for the transonic tables.

5.6 Link 50 Subroutines

This overlay and subroutines contained within are the essence of the TD2P module. The majority of this section is from Reference 4 with the major modifications restricted to Sections 5.6.2 through 5.6.12. However, almost all of the subroutines have been modified to some extent in order to bring this calculational module upto date. That is, this calculational module has been updated from FØRTRAN II to FØRTRAN IV by the extensive use of labeled common blocks and the reductions in the amounts of equivalences to blank common areas.

Other modifications made to the TD2P module include the extensive transmittal of data to reduce the amount and complexity of data needed to execute this module.

5.6.1 Program LINK50

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to other computer systems. The only function of this routine is to call subroutine TD2P.

5.6.2 Subroutine DIST

Input: $\bar{D}$, N, σ_g

Output: D_j, $\frac{m(D_j)}{m_T}$ $\qquad j=1, \ldots N$

Description :

Given an arbitrarily large number of spherical drops of mass mean diameter, $\bar{D}$, which are assumed to possess a logarithmic normal distribution about $\bar{D}$, then the mass density is given by:

$$\frac{d\left(\frac{m(D)}{m_T}\right)}{d\,D} = \left[(2\pi)^{1/2}\, D \ln \sigma_g\right]^{-1} \exp\left[-\frac{1}{2}\left(\frac{\ln \frac{D}{\bar{D}}}{\ln \sigma_g}\right)^2\right]$$

where

D is particle diameter

m_T is the total particle mass

σ_g is the geometric standard deviation

It follows that

$$m_T = m_T \int_0^{\infty} \frac{m(D)}{m_T}\, dD.$$

The purpose of this subroutine is to determine a set of N drop diameters

$$D_j \qquad j=1, \ldots N$$

such that

$$\int_{D_{j-1}}^{D_j} \frac{m(D)}{m_T}\, dD = \frac{2j-1}{2N} \qquad , D_o \equiv 0$$

The method described above is the same as presented in References 10 and 11.

5.6.3 Function EISP

Function calculates the error function

$$EISP = \epsilon - \epsilon(x)$$

used in finding the one dimensional - zero lag I_{sp} as a function of area ratio, ϵ. The independent variable, x, is either the gas temperature (before or after solidification of the condensed phase) or the fractional amount of solidified condensed phase (during solidification). The I_{sp} corresponding to the independent variable is also calculated.

This function is called by subroutine SEEK which is called by subroutine ISPID.

5.6.4 Function FAM

This function calculates the error function

$$FAM(M) = \epsilon - \epsilon(M)$$

where ϵ = requested area ratio

$\epsilon(M)$ = area ratio at the Mach number, M, given as the argument of function FAM.

The FAM function is called by subroutine SEEK (Section 5.6.8) which is called by TBLEDG (Section 5.6.9) to determine the subsonic-transonic Mach numbers as a function of area ratio for communication to the TBL module.

The error function is calculated using the following:

$$A^*/A = ASA = \frac{\gamma+1}{2} M \left(1 + \frac{\gamma-1}{2} M^2\right)^{2(\gamma-1)/(\gamma+1)}$$

$$FAM = \epsilon - 1.0/ASA$$

5.6.5 Subroutine FIND

The purpose of this subroutine is to perform linear or quadratic interpolation from a table.

Calling Sequence

CALL FIND (N,T,X,Y)

where

N is a flag such that:

N = 0 implies linear interpolation
N ≠ 0 implies quadratic interpolation

T is a table of the format specified below
X is the argument
Y is the result

Usage

The table, T, must be a column of a FORTRAN array. The format of the table is:

T(1)	a temporary cell used by FIND must be 0. initially.
T(2)	minimum argument
T(3)	function of minimum argument
.	
.	
.	
T(2n)	nth (maximum) argument
T(2n+1)	function of the nth argument
T(2n+2)	0. signifying the end of table
T(2n+3)	0. signifying the end of table

If FIND is entered with an argument above the table maximum or below the table minimum, it will store the function of the maximum or minimum argument, respectively. If FIND is entered with an argument which is between the first or last two table arguments, linear interpolation will be used.

Functional Description

This subroutine saves its place in a table and, on an option, either interpolates by the linear interpolation formula:

$$y = \frac{y_2 - y_1}{x_2 - x_1} (x-x_1) + y_1 \qquad x_1 < x \leq x_2$$

or performs a four point quadratic interpolation by the formula:

$$y = A(x-x_1)^2 + B(x-x_1) + C \qquad x_0 < x_1 < x \leq x_2 < x_3$$

where

$$d = x_2-x_1$$

$$e = x_1-x_0$$

$$f = x_3-x_1$$

$$A = \frac{(y_2 - y_1)(e-f)/d + y_0 - 2y_1 + y_3}{d(e-f) + e^2 + f^2}$$

$$B = (y_2-y_1)/d - Ad$$

$$C = y_1$$

The above quadratic formula is constrained such that the function y(x) passes through points (x_1, y_1) and (x_2, y_2) and has the property

$$[y(x_0)-y_0] - [y(x_3) - y_3] = 0$$

Note: There is, depending on the computer system used, a possible entry point name conflict between this routine and the subroutine of Section 5.1.6 by the same name. These two routines are not the same. See Appendix A for conversion aides.

5.6.6 Subroutine ISPID

This subroutine calculates I_{sp} for a given area ratio assuming one dimensional equivalent perfect gas flow with zero velocity and temperature lags between the gas and condensed phases. This I_{sp} is referred to as $I_{sp_{1DO_\ell}}$ in most of the documentation concerning the SPP code.

The calling sequence to ISPID is CALL ISPID(E,ISP,XI)

where E = area ratio, ϵ, at which the $I_{sp_{1DO_\ell}}$ is desired

ISP = $I_{sp_{1DO_\ell}}$ on exit from ISPID

XI = initial guess at the independent variable

Before and after solidification of the condensed phase, the gas temperature is used as the independent variable to iterate on the solution, however, during solidification, the fractional amount of solid to total condensed phase is used.

This routine utilizes subroutines SEEK (Section 5.6.8) and EISP (Section 5.6.3) to determine the root of the appropriate equations.

The constants used in these routines are calculated in subroutine AGP (Section 5.6.12).

5.6.7 Subroutine SEEK

This subroutine utilizes subroutine ITER to find the root of the equation:

$$f(x) = 0$$

Subroutine ITER predicts new guesses of the root, x_N, given old values of the error, f_o, and guess at the root, x_o, using the method of secants. The SEEK subroutine attempts to take maximum advantage of the properties of the secant method. (See the write up on subroutine ITER).

Calling Sequence

Call SEEK (XANS, XØ, MAXIT, EPSI, FCALC, IB, B, IT, IERR)

where

XANS	=	the root of f(x)=0 or the last guess for x if the maximum number of iterations has been exceeded.
XØ	=	initial guess at the value of the root.
MAXIT	=	maximum number of iterations allowed in finding the root.
EPSI	=	a solution to f(x)=0 is claimed if $\|f(x)\| <$ EPSI or $\|\Delta x\| <$ EPSI/100.
FCALC	=	an external function subprogram which calculates the error, f(x). The calling sequence is F=FCALC(x).
IB	=	a bounding flag such that if
		IB = 0 no bounding.
		IB = 1 the root is assumed to lie within the values of B(1) and B(2).
		IB = -1 the maximum absolute change allowed between successive guesses of the root is constrained to be no larger than $\|B(1)*\|x_o\| + B(2)\|$
B(I)	=	a vector of length 2. The use of this vector is described above.
IT	=	the number of iterations required to obtain a solution.
IERR	=	a flag such that if
		IERR=0 a solution was obtained.
		IERR=1 failed to converge on the root in MAXIT iterations.

5.6.8 Subroutine TBLEDG

This subroutine calculates the subsonic-transonic edge conditions for the trubulent boundary layer module, TBL. The number of points, NTBL, and the initial location of the subsonic start conditions, XITBL, are determined by user input in the $TD2 namelist (Volume III, Section 2.9).

The method used in constructing the edge tables for TBL is as follows:

1. The region from XITBL to the MOC initial line is broken up into NTBL equally spaced points.
2. Subroutine SEEK and function FAM are used to calculate the 1-D equivalent perfect gas Mach number, M_{1D}, at the area ratio where the initial line intersects the wall, $\epsilon_{i\ell}$.
3. The axial position, Z_o, in the nozzle which corresponds to an inlet area ratio of nine is located.
4. The edge Mach numbers are then calculated from

$$M_e = M_{1D}\left\{1 + \frac{Z-Z_o}{Z_{i\ell}-Z_o} \cdot \frac{M_{i\ell}-M_{1Di\ell}}{M_{1Di\ell}}\right\}$$

where the subscripts

$i\ell$ = initial line

1D = 1 dimensional Mach numbers calculated using SEEK and FAM as a function of axial position, Z.

In addition to the above, subroutine TBLEDG also recalculates the stagnations pressure and temperature communicated to the TBL module. The values of P_o and T_o are calculated from the static P and T at the initial line wall point in order that edge conditions calculated by TBL would more closely resemble the values calculated by TD2P.

5.6.9 Subroutine TD2P

Subroutine TD2P is the master subroutine of the TD2P module. It controls the initialization of variables, stores in the proper locations values transmitted from other modules, reads in user input in the $TD2 namelist, and directs the flow of the computation.

In the case that particle sizes are not input to the module, TD2P calculates the mean particle size (in microns) as

$$\bar{d}_p = XK \cdot PC^{PEXP} \cdot (\xi)^{XIEXP} (1 - EXP(XL*XLSTAR)) *(1 + 24 \cdot XD \cdot r^*)$$

$$\xi = 100\ (WPWGT/((1 + WPWGT) \cdot XMLW))$$

where XK, PEXP, XIEXP, XL, and XD are constants set in DATA statements or they may be input in the $TD2 namelist.

PC = chamber pressure (psia)

XLSTAR = L* (in)

ξ = moles of condensed phase/100gm.

WPWGT = $\dot{m}$ gas/$\dot{m}$ condensed phase

XMLW = molecular weight of the condensed phase

The particle size distribution is then calculated by calling subroutine DIST.

Overlay 5, 1 (LINK51) is called to calculate the average gas properties used by the rest of the TD2P routines (see subroutine AGP, Section 5.6.12).

The following items are then computed and made available:

$$\rho_{g_o} = \frac{P_{g_o}}{R\ T_{g_o}}$$

$$\rho_{p_o} = \rho_{g_o}\left(\frac{\Sigma \dot{w}_{p_j}}{\dot{w}_g}\right)$$

$$\left(\frac{\dot{w}_{p_j}}{\dot{w}_g}\right) = \left(\frac{\Sigma \dot{w}_{p_j}}{\dot{w}_g}\right)\left(\frac{\dot{w}_{p_j}}{\Sigma \dot{w}_{p_j}}\right) \quad ; \qquad j=1,\ldots,j_{max}$$

$$k_{equil} = \gamma \frac{1 + \frac{\Sigma \dot{w}_{p_j}}{\dot{w}_g} \frac{c_{p_g}}{c_{p_\ell}}}{1 + \frac{\Sigma \dot{w} P_j}{\dot{w}_g} \frac{c_{p_\ell}}{c_{p_g}} \gamma}$$

$$b = \frac{1 + \frac{\Sigma \dot{w}_{p_j}}{\dot{w}_g}}{1 + \frac{\Sigma w_{p_j}}{\dot{w}_g} \frac{c_{p_\ell}}{c_{p_g}}}$$

$$\dot{w}_{g_{equil}} = \rho_{g_o} \left(\frac{2}{k_{equil} + 1} \right) \left(\frac{1}{k_{equil} - 1} \right) \cdot \left(\frac{2 c_{p_g} T_{g_o} (k_{equil} - 1)}{b \, (k \, equil + 1)} \right)^{1/2} \pi r^{*2}$$

$$a_{oc} = \frac{9}{2} \frac{\mu_{g_o} r^*}{m_p}$$

$$c_g = \frac{3.42}{\gamma^{1/2} P_r}$$

Overlay 5,2 (LINK52) is then called for execution of the Axisymmetric Subsonic-Transonic Subprogram and then the following items are computed and made available:

$$C_c = \frac{\mu_{g_o}}{(T_{g_o} R)^N}$$

$$b_{p_j} = \frac{3C_{p_g}}{P_r m_p} \quad \frac{\mu_{g_o}}{(T_{g_o} R)^N} \quad \frac{r^*}{r^2_{p_j}} \quad ; \quad j=1,\ldots,j_{max}$$

after which Overlay 5,3 (LINK53) is called for execution of the Axisymmetric Supersonic Method of Characteristics Subprogram.

5.6.10 Program LINK51

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to the computer systems. The only function of this routine is to call subroutine AGP.

5.6.11 Subroutine AGP

The purpose of this subroutine is to calculate average gas properties based on the results of a one-dimensional calculation using variable thermodynamic properties (usually, the results of an ØDE calculation). These properties are then used in the calculation of one-dimensional and two dimensional I_{sp} in the rest of the TD2P module. The method used is outlined below.

Using Newton's method, an expansion coefficient, γ, is found such that a polytropic expansion from the chamber temperature, T_{go}, to the solidification temperature, T_{pm}, will occur at the input expansion ratio, ϵ_{pm}. The iteration equations used are

$$\bar{\gamma}^{(i+1)} = \bar{\gamma}^{(i)} - \left(\frac{\epsilon - \epsilon_{pm}}{\epsilon}\right) \epsilon \Big/ \frac{d\epsilon}{d\bar{\gamma}}$$

where

$$\epsilon = \left(\frac{(\bar{\gamma} - 1)\, T_{pm}}{2(T_{go} - T_{pm})}\right)^{1/2} \left(\frac{2}{\bar{\gamma} + 1} \frac{T_{go}}{T_{pm}}\right)^{\frac{\bar{\gamma}+1}{2(\bar{\gamma}-1)}}$$

and

$$\frac{\epsilon}{\frac{d\epsilon}{d\bar{\gamma}}} = -\frac{(\gamma - 1)^2}{\ln\left(\frac{2}{\bar{\gamma} + 1} \frac{T_{go}}{T_{pm}}\right)}$$

Next the value of the specific heat, C_{pg}, for the gas phase is calculated from the one dimensional energy equation as

$$C_{pg} = \frac{U^2_{gm}}{2(T_{go} - T_{pm})} \left(1 + \frac{\dot{w}_p}{\dot{w}_g}\right) - \frac{\dot{w}_p}{\dot{w}_g} C_{pl}$$

i.e. the reference enthalpy at stagnation is taken as zero. The flow velocity at the T_{pm} is given. It is also assumed that C_{pl} for $T \leq T_{pm}$ is also known.

An expansion coefficient for the gas phase is next calculated assuming no lag:

$$\gamma = \frac{\bar{\gamma}}{1 - (\bar{\gamma} - 1)\dfrac{\dot{w}_p}{\dot{w}_g}\dfrac{C_{pl}}{C_{pg}}}$$

so that

$$R = \frac{\gamma - 1}{\gamma} C_{pg}$$

Next the value of C_{ps}, the particle specific heat after solidification ($T < T_{pm}$), is calculated from the energy equation as

$$C_{ps} = \frac{\dot{w}_g}{\dot{w}_p (T_{pm} - T_{ps})} \left[\frac{1}{2} \left(1 + \frac{\dot{w}_p}{\dot{w}_g} \right) U_{gs} - C_{pg} (T_{go} - T_{ps}) - \frac{\dot{w}_p}{\dot{w}_g} C_{pl} (T_{go} - T_{pm}) - \frac{\dot{w}_p}{\dot{w}_g} \Delta H_m \right]$$

The values of T_{ps} and U_{gs} are input values corresponding to some point after solidification. The above value of C_{ps} is then checked against the actual value of C_{ps} at the solidification temperature. If the calculated value is more than 10% different than the actual value, the actual value is used. The reasons for the above test are

a) it prevents ridiculous values of C_{ps} from being used

b) it prevents the results of the TD2P calculation from being too sensitive to the area ratio selected at which values of T_{ps} and U_{gs} are obtained

If the Prandtl number, Pr, is not input or transmitted from the ØDE module, it is calculated, using kinetic theory, as

$$Pr = 4\gamma/(9\gamma - 5)$$

5.6.12 Program LINK52

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to other computer systems. The only function of this routine is to call subroutine PARTIL.

5.6.13 Subroutine ABCALC

Coefficients are calculated for first and second order transonic approximations:

First Order;

$$C_o = u_o^2 - 1$$

$$C_1 = 1 - \frac{k-1}{k+1} u_o^2$$

$$a_{20} = \frac{C_o \alpha}{4C_1}$$

$$a_{21} = \frac{u_o \alpha^2}{2C_1}$$

$$a_{40} = \frac{u_o a_{21} \alpha}{8C_1}$$

Second Order;

$$A_{201} = 2 \frac{k-1}{k+1} u_o \alpha$$

$$b_{21} = \frac{4A_{201} a_{20} + C_o \beta}{4C_1}$$

$$A_{102} = \alpha^2 + u_o \beta$$

$$b_{22} = \frac{2u_o \alpha\beta + A_{102} \alpha + 4A_{201} a_{21}}{4C_1}$$

$$A_{120} = 2u_o b_{21}$$

$$A_{220} = 2 \frac{k-1}{k+1} u_o a_{21}$$

$$A_{320} = \frac{8}{k+1} u_o a_{20}$$

$$b_{40} = \frac{2C_o b_{22} + A_{120} \alpha + 4A_{220} a_{20} + 2A_{320} a_{21}}{16C_1}$$

$$A_{121} = 2a_{21} \alpha + 4u_o b_{22}$$

$$A_{321} = \frac{8}{k+1} u_o a_{21}$$

$$b_{41} = \frac{2u_o a_{21}\beta + 4u_o b_{22}\alpha + A_{121}\alpha + 4A_{220}a_{21} + 16A_{201}a_{40} + 2A_{321}a_{21}}{16C_1}$$

$$A_{140} = a_{21}^2 + 2u_o b_{41}$$

$$A_{340} = \frac{16}{k+1} u_o a_{40}$$

$$b_{60} = \frac{4u_o a_{21} b_{22} + A_{140}\alpha + 16A_{220} a_{40} + 2A_{340} a_{21}}{36C_1}$$

5.6.14 Subroutine CCALC

Coefficients are calculated for the third order transonic approximations:

Third Order;

$$B_{202} = \frac{k-1}{k+1} A_{102}$$

$$C_{22} = \frac{C_o \gamma/2 + 4A_{201} b_{21} + 4B_{202} a_{20}}{4C_1}$$

$$B_{103} = u_o \frac{\gamma}{3} + \alpha\beta$$

$$C_{23} = \frac{4A_{201} b_{22} + 4B_{202} a_{21} + A_{102}\beta + B_{103}\alpha + u_o\alpha\gamma}{4C_1}$$

$$B_{120} = 4 \frac{k-1}{k+1} a_{20}^2$$

$$B_{220} = \frac{k-1}{k+1} A_{120}$$

$$C_{40} = \frac{B_{120}\alpha + 4B_{220} a_{20} + 2A_{320} b_{21} + 2C_o C_{22}}{16C_1}$$

$$B_{121} = 4u_o C_{22} + 2b_{21}\alpha + 8 \frac{k-1}{k+1} a_{20} a_{21}$$

$$B_{221} = \frac{k-1}{k+1} A_{121}$$

$$B_{321} = \frac{8}{k+1} (a_{20}\alpha + u_o b_{21})$$

$$C_{41} = \frac{1}{16C_1} (4A_{320} b_{22} + 2A_{321} b_{21} + 16A_{201} b_{40} + 4A_{220} b_{21}$$

$$+ B_{121}\alpha + 4B_{221} a_{20} + 4B_{220} a_{21} + 2B_{321} a_{21} + A_{120}\beta$$

$$+ 6C_o C_{23} + 4u_o C_{22}\alpha)$$

$$B_{122} = 6u_o C_{23} + a_{21}\beta + 4b_{22}\alpha + 4 \frac{k-1}{k+1} a_{21}^2$$

$$B_{322} = \frac{8}{k+1} (a_{21}\alpha + u_o b_{22})$$

$$C_{42} = \frac{1}{16C_1}\left(B_{122}\alpha + 4B_{221}a_{21} + 16B_{202}a_{40} + 2B_{322}a_{21} + A_{121}\beta\right.$$

$$\left. + 2A_{102}b_{22} + 16A_{201}b_{41} + 4A_{220}b_{22} + 4A_{321}b_{22}\right.$$

$$\left. - 12u_oC_{23}\alpha + u_oa_{21}\gamma\right)$$

$$B_{140} = 2u_oC_{41} + 2a_{21}b_{21} + 16\frac{k-1}{k+1}a_{20}a_{40}$$

$$B_{240} = \frac{k-1}{k+1}A_{140}$$

$$B_{340} = \frac{8}{k+1}\left(a_{20}a_{21} + 2u_ob_{40}\right)$$

$$C_{60} = \frac{1}{36C_1}\left(B_{140}\alpha + 4B_{240}a_{20} + 16B_{220}a_{40} + 2B_{340}a_{21} + 2A_{120}b_{22}\right.$$

$$\left. + 16A_{220}b_{40} + 2A_{340}b_{21} + 4A_{320}b_{41} + 2C_oC_{42} + 4u_oa_{21}C_{22}\right)$$

$$B_{141} = 4a_{21}b_{22} + 2b_{41}\alpha + 16\frac{k-1}{k+1}a_{21}a_{40} + 4u_oC_{42}$$

$$B_{341} = \frac{8}{k+1}\left(a_{21}^2 + 2a_{40}\alpha + 2u_ob_{41}\right)$$

$$C_{61} = \frac{1}{36C_1}\left(4A_{340}b_{22} + 4A_{321}b_{41} + 36A_{201}b_{60} + 16A_{220}b_{41}\right.$$

$$\left. + B_{141}\alpha + 4B_{240}a_{21} + 16B_{221}a_{40} + 2B_{341}a_{21} + A_{140}\beta\right.$$

$$\left. + 2A_{121}b_{22} + 4u_oC_{42}\alpha + 12u_oa_{21}C_{23}\right)$$

$$B_{160} = 2u_oC_{61} + 2a_{21}b_{41} + 16\frac{k-1}{k+1}a_{40}^2$$

$$B_{360} = \frac{8}{k+1}\left(2a_{21}a_{40} + 3u_ob_{60}\right)$$

$$C_{80} = \frac{1}{64C_1}\left(B_{160}\alpha + 16B_{240}a_{40} + 2B_{360}a_{21} + 2A_{140}b_{22} + 36A_{220}b_{60}\right.$$

$$\left. + 4A_{340}b_{41} + 4u_oa_{21}C_{42}\right)$$

5.6.15 Subroutine DCALC

A dummy subroutine containing only a RETURN statement. This subroutine is reserved for the calculation of coefficients for the fourth order transonic approximations.

5.6.16 Subroutine FCALC

This subroutine calculates the boundary condition functions, f_i, from coefficients and derivatives determined by subroutines ABCALC, CCALC, and PCALC:

$$f_1 = u - u_o$$

$$f_2 = u_o \frac{dr}{dx} - v$$

$$f_3 = \frac{du}{dx}\frac{dr}{dx} + u_o \frac{d^2r}{dx^2} - \frac{dv}{dx}$$

$$f_4 = \frac{d^2u}{dx^2}\frac{dr}{dx} + 2\frac{du}{dx}\frac{d^2r}{dx^2} + u_o \frac{d^3r}{dx^3} - \frac{d^2v}{dx^2}$$

$$f_5 = \frac{d^3u}{dx^3}\frac{dr}{dx} + 3\frac{d^2u}{dx^2}\frac{d^2r}{dx^2} + 3\frac{du}{dx}\frac{d^3r}{dx^3} + u_o \frac{d^4r}{dx^4} - \frac{d^3v}{dx^3}$$

If a special throat expansion (see Step 3, subroutine PARTIL) is being performed, the following variables are computed:

$$u^* = u_o + \alpha x_o + \beta \frac{x_o^2}{2} + \gamma \frac{x_o^3}{6}$$

$$\alpha^* = \alpha + \beta x_o + \gamma \frac{x_o^2}{2}$$

$$\gamma^* = \frac{4}{z_s}\left[u^* - \frac{u_s}{u^*} + \frac{z_s}{2}\left(\alpha^* + \frac{(u_z)_s}{u^*}\right)\right]$$

$$\beta^* = \frac{1}{z_s}\left[\frac{(u_z)_s}{u^*} - \alpha^* - \gamma^* \frac{z_s^2}{2}\right]$$

If transonic conditions in the nozzle inlet are to be fared to the throat conditions, the following variables are computed:

$$u_o = u^* + \alpha^* (z_w + x_o) + \beta^* \frac{(z_w + x_o)^2}{2} + \gamma^* \frac{(z_w + x_o)^3}{6}$$

$$\alpha = \alpha^* + \beta^* (z_w + x_o) + \gamma^* \frac{(z_w + x_o)^2}{2}$$

5.6.17 Subroutine JAMES

This subroutine provides control for the perfect gas transonic flow approximation computations. Newton's Method is employed to find solutions (see subroutine NEWT)

x_o , sonic point displacement

u_o , axial velocity/critical flow velocity

α , 1st order coefficient

β , 2nd " "

γ , 3rd " "

to the non-linear equations (see subroutine FCALC)

$f_1 = o$

$f_2 = o$

$f_3 = o$

$f_4 = o$

$f_5 = o$

Solutions are sought such that:

if 1st order then β, γ, $f_4 = o$, and $f_5 = o$ are deleted

if 2nd order then γ and $f_5 = o$ are deleted

if the conical inlet angle, θ_i, is greater than the faring angle, θ_f, then a special expansion is performed at the nozzle throat with u_o and $f_1 = o$ deleted. Expansions at points $(r_w, z_w)=1$ upstream of the faring point are performed with u_o and α computed by subroutine FCALC and f_5 set zero.

Calling Sequence

RR	r_w, radial wall position coordinate
XX	z_w, axial wall position coordinate
TT	$1/R_c$, reciprocal of the throat radius of curvature
VEK	k, the average gas-particle expansion coefficient

5.6.18 Subroutine LEGS

Purpose

To solve an overdetermined linear system of equations $AX = b$ in the least squares sense, i.e., to solve the "normal equations" $A^TAX = A^Tb$.

The superscript T denotes the transposed matrix.

Usage

Calling Sequence:

```
CALL LEGS(AB,X,BNDS,M,N,MD,ND,ATA,R,Z,SUS,SUSP,I1,I2,I3,I4,I5,IDLE)
```

where

AB = the M by N variably dimensioned augmented matrix (A,b) with maximum row dimension MD and maximum column dimension ND.

X = the solution vector of dimension N.

BNDS = the bounds vector of dimension N.

M = the number of rows in the augmented matrix (A,b).

N = the number of columns in matrix A.

MD = the row dimension of array AB. MD ≥ M.

ND = the column dimension of array AB. ND ≥ N + 1.

ATA = a one dimensional array in which the upper triangular part of the augmented normal matrix

$$\begin{pmatrix} A^TA & A^Tb \\ b^TA & b^Tb \end{pmatrix}$$

is stored by rows. The dimension of this array must be greater or equal to $(N+1)(N+2)/2$.

R = a one dimensional array used for storage. If the inverse of A^TA is requested, the lower triangular part is stored by rows in this array. The dimension must be greater than or equal to $((N+2)(N+3)/2) - 1$.

Z = an output indicator. If the bounding option is chosen, $Z \neq 0$ indicates that the solution was affected by the bounds.

SUS = the squared norm of b, i.e., SUS = $||b|| = b^Tb$

SUSP = the squared norm of AX-b, i.e., SUSP = $||AX-b||^2$

I1 = Bounding option. If I1 = 0 the solution is unbounded and the routine minimizes $||AX-b||^2$. For I1 $\neq$ 0, the routine minimizes $||AX-b||^2$ under the side condition that

$$\sum (X_i/B_i)^2 \leq 1.$$

The sum extends over all i for which $B_i > 0$. If $B_i = 0$, the routine ignores the i'th row and column of the augmented normal matrix and sets $X_i = 0$. If $B_i < 0$, the routine does not bound X_i. The constants B_i must be stored in BNDS. When I1 = 0, the BNDS vector is set to -1.

I2 = Solution option. If I2 = 0, the routine solves for X. If I2 $\neq$ 0, the BNDS vector is placed into the X vector and SUS, SUSP, and $(A^TA)^{-1}$ (if requested) are computed. This may be used for checkout purposes when the routine is used in an iterative process for solving nonlinear systems. It is also used for comparing two "solutions" of a nearly singular system.

I3 = ATA option. If I3 = 0, the augmented normal matrix is computed and placed in ATA. If I3 > 0, the normal matrix computation will be omitted. It is assumed the user has already calculated this matrix and placed it in ATA. If I3 < 0, the normal matrix is computed and added to whatever is in ATA. This is helpful in case the augmented matrix must be partitioned.

I4 = Inverse option. If I4 $\neq$ 0, the normal matrix is <u>not</u> inverted. Upon exit R will contain

a. the lower triangular part of the matrix S, stored by rows, where S is an Nth order lower triangular matrix with (-1) on the main diagonal

b. the solution vector X

c. -1

d. the main diagonal of the diagonal matrix D.

Let $C = A^TA + zB^{-2}$, then S and D have the property that $SCS^T = D^*$

If I4 = 0, the matrix C is inverted as $C^{-1} = S^TD^{-1}S$ and C^{-1} replaces S before exit from the routine.

I5 = ATA only option. If I5 $\neq$ 0, the routine exits after forming the normal matrix. If I5 = 0, the routine continues by checking I2.

* A more complete explanation is found under Functional Description.

IDLE = ATA row deletion option. In normal use IDLE is a single cell integer of value 0 and has no effect on the solution. The use of this option is best explained by an example. Suppose the user forms the augmented matrix and wishes to delete certain rows from consideration in the least squares problems (because of bad data or some other reason). He wishes to delete rows 20, 31-35, 5 and 14-16. This can be accomplished by putting the following sequence of integer into IDLE: 20,0,31,35,5,0,14,16,0

The list is terminated by a zero in an odd position. IDLE must be dimensioned in the driving program if it is to be used.

Functional Description

1. Bounds - When the bounds option is chosen the program minimizes the quadratic form $||AX-b||^2$ subject to the side condition $(B^{-2}X.X) \leq 1$, where B^{-2} is the diagonal matrix whose i'th diagonal element is B_i^{-2} if $B_i > 0$, whereas if $B_i < 0$, the i'th diagonal element is equal to zero (If $B_i = 0$, the routine ignores the i'th row and column of the augmented normal matrix and sets $X_i = 0$).

   The method of Lagrange multipliers is applied to solve this problem Thus, the system of linear equations

   $$(A^TA + zB^{-2})\, X = A^Tb$$

   is solved for various choices of the real positive multiplier z, until a solution $X(z)$ is found which satisfies the side conditions.

   The search for an appropriate value of z is systematized as follows: With $\epsilon_1 = 1/10$ and $\epsilon_2 = 2/10$, the routine

   a. Finds $X = X(0)$. If $(B^{-2} X, X) \leq 1 + \epsilon_1$, the problem is considered solved. Otherwise,

   b. define $y(z) = (B^{-2} X(z), X(z))$. Now, $y(0) > 1 + \epsilon_1$. Compute $y(z_1), y(z_2), \ldots$, where $z_j = z_{j-1} + 10(j-1)H$ and $H = \underset{i}{MAX}$ $[B_i^2(|A^Tb)_i|/B_i - A^TA(i,i))]$ whenever this expression is positive, otherwise $H = \underset{i}{MAX}\ [.0001\ A^TA(i,i)B_i^2]$

   The computation of the $y(z_j)$'s is continued until a z is found such that $1 = \epsilon_2 \leq y(z_j) \leq 1 + \epsilon_1$ in which case $X(z_j)$ is accepted as the solution, or until two values of z_j, renamed z_1 and z_2 are found such that $y(z_1) > 1 + \epsilon_1$ and $y(z_2) > 1 - \epsilon_2$.

   The required value of z is now bracketed. Now

   c. a value of z_3 is chosen with $z_2 < z_3 < z_1$. If $1 = \epsilon_2 \leq y(z_3) \leq 1 + \epsilon_1$ $X(z_3)$ is accepted as the solution. Otherwise,

d. inverse quadratic interpolation to zero is applied to obtain a new estimate z_4. If $1 - \epsilon_2 < y(z_4) \leq 1 + \epsilon_1$, $X(z_4)$ is accepted as the solution. Otherwise,

e. select from the set z_1, z_2, z_3 and z_4 that pair which bracket the solution most tightly and return to (c).

The routine will halt this iterative process if the number of solutions of the linear system reaches twenty.

2. Solution of the linear system - Let $C = A^TA + zB^{-2}$. The routine finds two matrices S and D such that $SCS^T = D$, where D is diagonal and S lower triangular with minus ones on the main diagonal. It is easy to find S and D for a 1 x 1 matrix C. Suppose they have been found for a k x k matrix C. Now augment C by another row and column

$$\begin{pmatrix} C & d \\ d^T & \alpha \end{pmatrix}$$

and seek a vector ω and a scalar β such that

$$\begin{pmatrix} S & 0 \\ \omega^T & -1 \end{pmatrix} \begin{pmatrix} C & d \\ d^T & \alpha \end{pmatrix} \begin{pmatrix} S^T & \omega \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} D & 0 \\ 0 & \beta \end{pmatrix} .$$

It is easy to verify that $\omega = S^TD^{-1}Sd$

$$\beta = 2 - \omega^Td$$

satisfy the requirements. The routine builds the matrices S and D by the above process the final result being a decomposition of the augmented matrix.

$$\begin{pmatrix} S & 0 \\ \omega^T & -1 \end{pmatrix} \begin{pmatrix} A^TA+zB^{-2} & A^Tb \\ b^TA & b^Tb \end{pmatrix} \begin{pmatrix} S^T & \omega \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} D & 0 \\ 0 & \alpha \end{pmatrix}.$$

The vector ω is the solution vector. The above method is a variant of the "square root" method.

3. Residual Calculation - The residual is calculated as follows:

$$||AX - b||^2 = (A^TAX,X) - A(A^Tb,X) + (b,b) \quad .$$

5.6.19 Subroutine NEWT

Given a set of m functions in n unknowns ($m \geq n$)

$$f_i(x_1, x_2, \ldots, x_n) \qquad i = 1, \ldots m$$

this routine will attempt by Newton's method to find values

$$X_1, X_2, \ldots X_n$$

such that

$$\sum_{i=1}^{m} f_i^2(X_1, X_2, \ldots X_n)$$

is a minimum.

Restrictions

The user must supply initial values for the solution vector and also a subroutine to evaluate the functions. The subroutine must communicate with NEWT through CØMMØN statements. Subroutine TRW LEGS is required.

No error exits are provided.

Method

Newton's method is used to iterate for a solution vector. The matrix inversion is performed by subroutine TRW LEGS.

In the event Newton's method yields a vector which is farther from a solution in a least squares sense than the previous estimate, the increment vector is halved.

Usage

Calling Sequence:

```
CALL NEWT(M,F,MF,N,VAR,NVAR,FCALC,WF,PERTV,EPS1,MAXIT,
          NUMIT,BIN,AB,B,KB)
```

Index to Calling Sequence:

Input: M,MF,N,VAR,NVAR,WF,PERTV,EPS1,MAXIT,B,KB

Output: VAR,F,NUMIT

Working: BIN,AB

Explanation of Calling Sequence:

1) M is the total number of functions, m.

2) F is the array of the m function values found by FCALC.

3) MF is an array of m control words:

   if MF(I)=0, include F(I)
   if MF(I)=1, exclude F(I)

4) N is the total number of variables, n.

5) VAR is an array of the n variables, x_j $\quad j = 1,...n$

6) NVAR is an array of n control words:

   if NVAR(J)=0, include VAR(J)
   if NVAR(J)=1, exclude VAR(J)

7) FCALC is a name for the subroutine which calculates the functions, F(I). A program which calls NEWT must have an EXTERNAL statement containing this name.

8) WF is an array of m weighting factors. These are used in conjunction with EPS1 to determine whether a solution has been reached.

$$\omega_i = \text{WF(I)}$$

9) PERTV is a perturbation factor used in calculating the partial derivatives required by Newton's method. If PERTV = ε, then:

$$\frac{\partial F}{\partial x} = \frac{F(x + \delta) - F(x)}{\delta}, \text{ where } \delta = \max(|\varepsilon x|, \varepsilon * 10^{-3})$$

10) EPS1 is an error bound used to determine whether a solution has been reached. If ε_1 = EPS1, then a solution is claimed when

$$\sum_{i=1}^{m} (\omega_i F_i)^2 \leq \varepsilon_1$$

11) MAXIT is the maximum number of iterations allowed.

12) NUMIT is the number of iterations required for solution.

13) BIN is an array used internally by the routine and must be at least of dimension $n^2 + 7n + 3$.

14) AB is a two-dimensional array containing the augmented matrix of subroutine LEGS and must be at least of dimension AB(m, n+1).

15) B is an input bounds vector of dimension n. If the bounds option is chosen, the iteration increments ΔX_j are constrained to be less than B_j (see LEGS). B is not modified internally.

16) KB is an input flag to be set equal to 1 if the bounds option is desired. Otherwise, set KB = 0. (See LEGS.)

5.6.20 Subroutine ØNEL

The relations for one-dimensional two-phase perfect gas nozzle flow are integrated through the nozzle conical inlet (Region I of Figure 5.6-2) The coordinate system is centered at the apex of the conical inlet. The differential equations are:

$$\frac{dV_{p_j}}{dz'} = \frac{9}{2}\frac{\mu_g}{m_p}\frac{f'_{p_j}}{r^2_{p_j}}\frac{r^*\,(V_g - V_{p_j})}{V_{p_j}} \qquad j = 1, \ldots j_{max}$$

$$\frac{dT_{p_j}}{dz'} = \frac{-3\,\mu_g\, g'_{p_j}\, r^*}{m_p\, r^2_{p_j}}\frac{C_{p_g}}{P_r\, C_{p_\ell}}\frac{(T_{p_j} - T_g)}{V_{p_j}} \qquad j = 1, \ldots j_{max}$$

$$\frac{dV_g}{dz'} = \frac{V_g\,\gamma}{V_g^2 - \gamma RT_g}\left[\frac{2RT_g}{z'} + \frac{RC_{p_\ell}}{C_{p_g}}\sum_{j=1}^{j_{max}}\frac{\dot{w}_{p_j}}{\dot{w}_g}\frac{dT_{p_j}}{dz'}\right.$$

$$\left. - \sum_{j=1}^{j_{max}}\frac{\dot{w}_{p_j}}{\dot{w}_g}\left(V_g - \frac{R}{C_{p_g}}V_{p_j}\right)\frac{dV_{p_j}}{dz'}\right]$$

where*:

$$\rho_g = \dot{m}_g / V_g A$$

$$A = 2\pi r^{*2}(1 - \cos\theta_i)z'^2$$

$$T_g = T_{g_o} - \frac{V_g^2}{2C_{p_g}} - \frac{1}{C_{p_g}}\sum_{j=1}^{j_{max}}\frac{\dot{w}_{p_j}}{\dot{w}_g}\left[C_{p_\ell}(T_{p_j} - T_{g_o}) + \frac{V^2_{p_j}}{2}\right]$$

$$\mu_g = \mu_{g_o}\left(\frac{T_g}{T_{g_o}}\right)^N$$

---

* The area, A, is taken as a spherical cap of half angle θ_i and radius r*z'.

$$f'_{p_j} = f_{p_j}\left(\frac{1+2.52\,\bar{C}_j}{1+3.78\,\bar{C}_j}\right)$$

$$g'_{p_j} = f_{p_j}\left(1/\left(1+\frac{3.42\bar{C}_j\, f_{p_j}}{\gamma_g^{1/2}\, P_r}\right)\right)$$

$$f_{p_j} = f_p(R_{e_j}) \quad \text{(by interpolation from Table 5.6-1)}$$

$$R_{e_j} = 2\rho_g\, r_{p_j}(V_g - V_{p_i})/\mu_g$$

$$\bar{C}_j = \mu_g/\rho_g\, r_{p_j}\sqrt{RT_g}$$

The initial conditions are:

$$T_{p_{j_o}} = \frac{200}{199+\bar{\gamma}}\, T_{g_o}$$

$$\bar{\gamma} = 1 + (\gamma_g - 1)\,\frac{B}{C}$$

$$B = \frac{1 + \sum_{j=1}^{j_{max}} \frac{\dot{w}_{p_j}}{\dot{w}_g}}{1 + \frac{C_{P_\ell}}{C_{P_g}} \sum_{j=1}^{j_{max}} \frac{\dot{w}_{p_i}}{\dot{w}_g}}$$

$$C = 1 + [1 + (\gamma_g - 1)\,\frac{C_{P_\ell}}{C_{P_g}}\, B] \sum_{j=1}^{j_{max}} \frac{\dot{w}_{p_i}}{\dot{w}_g}$$

$$V_{g_o} = \left(\frac{\bar{\gamma}-1}{199+\bar{\gamma}}\right)^{1/2} V_{g_{max}}$$

$$V_{g_{max}} = \left(\frac{2C_{P_g}\, T_{g_o}}{B}\right)^{1/2}$$

$$V_{p_{j_o}} = V_{g_o}$$

R_{e_j}	f_p
0.	1.
1.25	1.
1.255	1.
1.26	1.001
1.265	1.002
1.582	1.063
1.995	1.141
2.51	1.224
3.16	1.315
3.98	1.412
5.01	1.517
6.31	1.625
7.95	1.745
10.	1.874
12.6	2.026
15.82	2.186
19.95	2.364
25.1	2.555
31.6	2.76
39.8	3.
50.1	3.252
63.1	3.534
79.5	3.825
100.	4.155
316.	7.9
1000.	20.
1001.	20.02
100000.	2000.

Table 5.6-1 The Particle Drag Table

and

$$z'_o = -\frac{1}{r^*}(A_o/2\pi(1 - \cos\theta_i))^{1/2}$$

$$A_o = \frac{\dot{m}_g}{\left(\frac{200}{199+\bar{\gamma}}\right)^{1/(\bar{\gamma}-1)} V_{g_o} \rho_{g_o}}$$

The integration proceeds until:

$$z' = z'_a$$

$$z'_a = -(1 + R_c(1 - \cos\theta_i))/\sin\theta_i$$

using an integration step size

$h = \Delta z_{min}$, which is the input item DZMIN

The one-dimensional gas-particle velocity lags, K_j, and thermal lags, L_j, are determined at position z'_a

$$K_j = \frac{V_{p_j}}{V_g}$$

$$L_j = \frac{T_{g_o} - T_{p_j}}{T_{g_o} - T_g}$$

The quantities

$$\frac{(u_z)s}{u^*} = \frac{dVg}{dZ'}/u^*$$

and

$$u_s/u^* = Vg/u^*$$

are also calculated to be used by subroutine FCALC.

5.6.21 Subroutine PARTIL

This subroutine is the master subprogram for the Subsonic-Transonic Link. The axisymmetric gas-particle flow field of Region II, Figure 5.6-1 is computed using the procedure outlined below.

STEP 1:

The variables u^*_{equil} and z'_s are computed;

$$u^*_{equil} = \left[\frac{2\, C_{P_g}\, T_{g_o}}{b} \left(\frac{k_{equil} - 1}{k_{equil} + 1} \right) \right]^{1/2}$$

$$z'_s = -R_c \sin \theta_i - \bar{R}(1 - \cos \theta_i)$$

$$\bar{R} = \left[1 + R_c (1 - \cos \theta_i) \right] / \sin \theta_i$$

initially

$$k = k_{equil}$$

STEP 2

Subroutine ØNE D is used to determine conditions at the upstream end ($z = z_s$) of Region II, Figure 5.6-1

Region II is next divided into zones as illustrated by the example given in Figure 5.6-2. Wall coordinates for the perturbation points are:

$$r_w = 1 + R_c (1 - \cos \theta_p)$$

$$z_w = R_c \sin \theta_p$$

where

$$\theta_p = -\theta_i, -\frac{n_i - 1}{n_i} \theta_i, \ldots 0, \frac{\theta_j}{n_j}, \ldots \theta_j$$

i.e. a total of $n_i + n_j + 1$ zones are constructed.

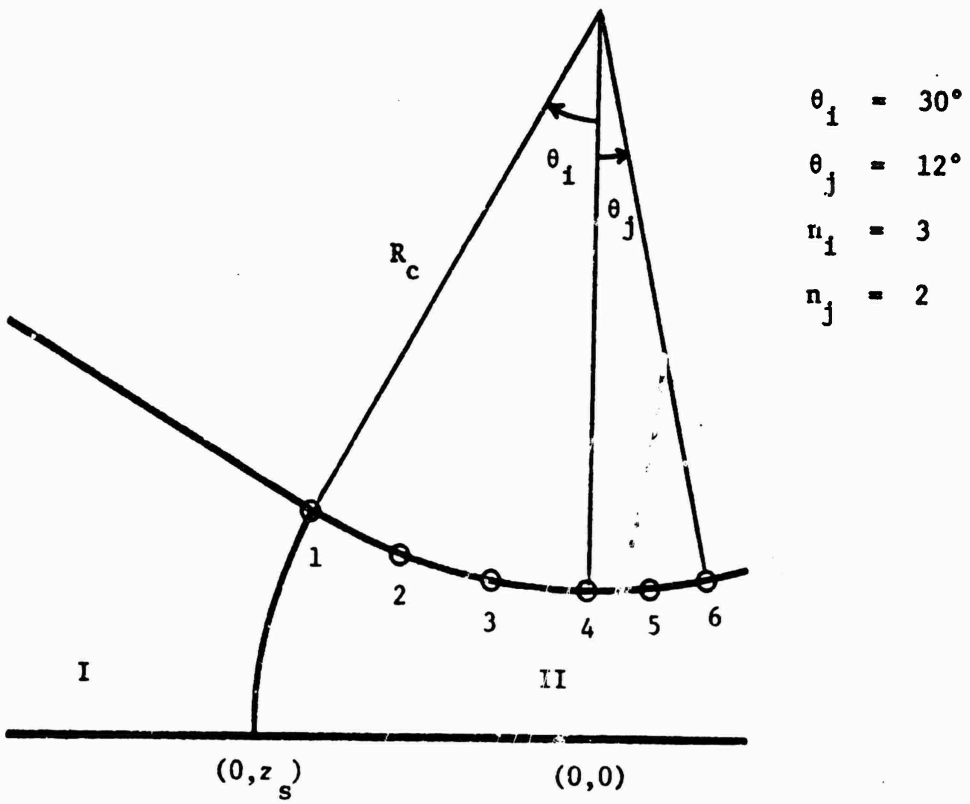


Figure 5.6-2 The Division of Region II into Zones

STEP 3:

The perfect gas transonic flow subroutine, JAMES, is used to determine the variables x_o, u_o, α, β, γ and the transonic approximation coefficients for each zone.

If faring is necessary ($\theta_i > \theta_f$) a special throat expansion is first performed and the variables u*, α*, γ* and β* are computed.

STEP 4:

One-dimensional gas-particle flow relations are integrated along the nozzle axis from z_s to z_{axis}. The differential equations integrated are those given in subroutine TRACE. Initial values for the gas variables T_g/T_{g_o} and u_g are evaluated using subroutine PRØP. Initial values for the particle variables are computed from K_j and L_j as determined by subroutine ØNE D. Transonic zones are exchanged as soon as

$$\frac{u_g}{u_g^*} > \frac{u_{oi} + u_{oi+1}}{2} .$$

Values K_j and L_j at z_{axis} are made available for use in Step 8.

STEP 5:

A corrected estimate (constant lag) for the expansion coefficient, k, is computed using values at $z = z_{axis}$

$$k = 1 + \frac{\gamma - 1}{1 + C_2 \gamma}$$

$$C_2 = \sum_{j=1}^{j_{max}} \frac{\dot{w}_{p_j}}{\dot{w}_g} \left[\frac{C_{P_\ell}}{C_{P_g}} L_j + K_j (1-K_j) C_1 \right]$$

$$C_1 = \frac{1 + \frac{C_{P_\ell}}{C_{P_g}} \sum_{j=1}^{j_{max}} \frac{\dot{w}_{p_j}}{\dot{w}_g} L_j}{1 + \sum_{j=1}^{j_{max}} \frac{\dot{w}_{Dj}}{\dot{w}_g} K_j^2}$$

the variable u_g^* is recomputed as

$$u_g^* = \sqrt{\frac{2\, C_{P_g}\, T_{g_o}}{b}\left(\frac{k-1}{k+1}\right)}$$

Using the above expansion coefficient, k, transonic conditions near the initial line are recomputed by subroutine JAMES. A corrected estimate for the nozzle mass flow, $\dot{m}_g$, is found by integrating gas properties along the initial supersonic data line using subroutine WDGI.

STEP 6:

Steps 2 through 5 are repeated twice to assure convergence in the variables k and $\dot{m}_g$.

STEP 7:

For $\theta_n = \frac{\theta_i}{20}, \frac{\theta_i}{3}, \frac{\theta_i}{2}$, and θ_i (n=1,..,4)

an initial position of

$$r_o = \bar{R} \sin \theta_n$$

$$z_o = z_s + \bar{R}\,(1-\cos \theta_n)$$

$$\bar{R} = \left[1 + R_c\,(1-\cos \theta_i)\right] \Big/ \sin \theta_i$$

is selected on the upstream boundary of Region II, Figure 5.6-2, and particle trajectories are integrated through Region II using subroutine TRACE.

Tables are constructed for the following variables as found along the initial line:

$$\psi_{j,n} = \frac{\dot{m}_{p_j}\,(1-\cos \theta_n)}{2\,\pi r^{*2}\,(1-\cos \theta_i)}$$

$$K_{j,n} = \frac{u_{p_j}}{u_g}$$

$$K'_{j,n} = \frac{v_{p_j}}{v_g}$$

$$L_{j,n} = \frac{T_{g_o} - T_{p_j}}{T_{g_o} - T_g}$$

$$r^2_{j,n}$$

where $n = 1,..4$

$j = 1,..j_{max}$

Limiting particle streamline trajectories correspond to n = 4.

STEP 8:

An initial supersonic data line is constructed consisting of NILP + 1 points with equal radial spacing. This data line extends from the axis to the wall along the parabola

$$z = z_{axis} - \left(\frac{z_{axis} - z_{wall}}{r^2_{wall}} \right) r^2.$$

Limiting particle streamline points are added along this data line and those equally spaced points which fall too close $\left(\text{within } \frac{r_{wall}}{2 \cdot NILP} \right)$ are deleted.

Gas properties P_g, ρ_g, u_g, and v_g are computed at each point using subroutine PRØP.

Particle properties u_{p_j}, v_{p_j}, h_{p_j}, and ρ_{p_j} are interpolated at each point from the tables constructed in Step 4 and Step 7. The interpolation formulas are:

$$u_{p_j} = K_j \, u_g$$

$$K_j = P(K_{j,n})$$

$$v_{p_j} = K'_j \, v_g$$

$$K'_j = P(K'_{j,n})$$

$$h_{p_j} = h_{p_\ell} + C_{p_\ell} \left(T_{p_j} - T_{p_m}\right)$$

$$T_{P_j} = T_{g_o} - (T_{g_o} - T_g) \, L_j$$

$$L_j = P(L_{j,n})$$

$$\rho_{p_j} = \frac{2 \psi_{j,1}/r_{j,1}^2 + 4a\left(\frac{\psi_{j,n}}{r_{j,n}^2}\right) r^2 + 6b \left(\frac{\psi_{j,n}}{r_{j,n}^2}\right) r^4 + 8c \left(\frac{\psi_{j,n}}{r_{j,n}^2}\right) r^6}{\left(u_{p_j} - v_{P_j} \frac{dz}{dr}\right)}$$

$$\frac{dz}{dr} = - \frac{2(z_{axis} - z_{wall})}{r_{wall}^2} r$$

where

$$P(x_{j,n}) = x_{j,o} + a(x_{j,n})\, r^2 + b(x_{j,n}) r^4 + c(x_{o,n}) r^6$$

$$c(x_{j,n}) = \frac{1}{(r_{j,4}^2 - r_{j,3}^2)} \left\{ \frac{(r_{j,4}^2 - r_{j,3}^2)(x_{j,2} - x_{j,o})}{r_{j,2}^2 (r_{j,3}^2 - r_{j,2}^2)(r_{j,4}^2 - r_{j,2}^2)} - \frac{(x_{j,3} - x_{j,o})}{r_{j,3}^2 (r_{j,3}^2 - r_{j,2}^2)} + \frac{(x_{j,4} - x_{j,o})}{r_{j,4}^2 (r_{j,4}^2 - r_{j,2}^2)} \right\}$$

$$b(x_{j,n}) = \frac{1}{(r_{j,3}^2 - r_{j,2}^2)} \left\{ \frac{x_{j,3} - x_{j,o}}{r_{j,3}^2} - \frac{x_{j,2} - x_{j,o}}{r_{j,2}^2} \right\} - c(x_{j,n})(r_{j,2}^2 + r_{j,3}^2)$$

$$a(x_{j,n}) = \frac{x_{j,2} - x_{j,o}}{r_{j,2}^2} - b(x_{j,n}) r_{j,2}^2 - c(x_{j,n}) r_{j,2}^4$$

The subscript n=o denotes axis values. The variables $K_{j,o}$ and $L_{j,o}$ are made available in Step 4 while $K'_{j,o}$ and $\psi_{j,o}$ are taken as:

$$K'_{j,o} = K'_{j,1}$$

$$\psi_{j,o} = \psi_{j,1}$$

STEP 9:

For each initial supersonic data line point the varables p_g, u_g, v_g, r, z, h_{p_j}, ρ_{p_j}, u_{p_j}, and v_{p_j} are printed.

5.6.22 Subroutine PCALC

The following quantities are computed:

$$x = x_o \quad , \quad r = r_w \quad , \quad x_w = z_w$$

$$\frac{dr}{dx} = x_w \left(R_c^2 - x_w^2\right)^{-1/2}$$

$$\frac{d^2r}{dx^2} = \left(R_c^2 - x_w^2\right)^{-1/2} + x_w^2 \left(R_c^2 - x_w^2\right)^{-3/2}$$

$$\frac{d^3r}{dx^3} = 3x_w \left(R_c^2 - x_w^2\right)^{-3/2} + 3x_w^3 \left(R_c^2 - x_w^2\right)^{-5/2}$$

$$\frac{d^4r}{dx^4} = 3 \left(R_c^2 - x_w^2\right)^{-3/2} + 18x_w^2 \left(R_c^2 - x_w^2\right)^{-5/2} + 15x_w^4 \left(R_c^2 - x_w^2\right)^{-7/2}$$

$$u' = \alpha x + a_{21} r^2$$

$$u'' = \beta \frac{x^2}{2} + (b_{21} + 2b_{22}x)\, r^2 + b_{41} r^4$$

$$u''' = \gamma \frac{x^3}{6} + (2c_{22}x + 3c_{23}x^2)\, r^2 + (c_{41} + 2c_{42}x)\, r^4 + c_{61} r^6$$

$$u = u_o + u' + u'' + u'''$$

$$v' = 2(a_{20} + a_{21}x)\, r + 4a_{40} r^3$$

$$v'' = 2(b_{21}x + b_{22}x^2)\, r + 4 (b_{40} + b_{41}x)\, r^3 + 6b_{60} r^5$$

$$v''' = 2(c_{22}x^2 + c_{23}x^3)\, r + 4 (c_{40} + c_{41}x + c_{42}x^2)\, r^3 + 6 (c_{60} + c_{61}x)\, r^5 + 8c_{80} r^7$$

$$v = v' + v'' + v'''$$

and

$$\frac{du}{dx} = u_x + u_r \frac{dr}{dx}$$

$$\frac{d^2u}{dx^2} = u_{xx} + 2u_{xr} \frac{dr}{dx} + u_r \frac{d^2r}{dx^2} + u_{rr} \left(\frac{dr}{dx}\right)^2 + 3u_{rrx} \left(\frac{dr}{dx}\right)^2 + u_{rrr} \left(\frac{dr}{dx}\right)^3 + 3u_{rr} \frac{dr}{dx} \frac{d^2r}{dx^2}$$

$$\frac{d^3u}{dx^3} = u_{xxx} + 3u_{xxr} \frac{dr}{dx} + 3u_{xr} \frac{d^2r}{dx^2} + u_r \frac{d^3r}{dx^3}$$

$$\frac{dv}{dx} = v_x + v_r \frac{dr}{dx}$$

$$\frac{d^2v}{dx^2} = v_{xx} + 2v_{xr} \frac{dr}{dx} + v_r \frac{d^2r}{dx^2} + v_{rr} \left(\frac{dr}{dx}\right)^2$$

$$\frac{d^3v}{dx^3} = v_{xxx} + 3v_{xxr} \frac{dr}{dx} + 3v_{xr} \frac{d^2r}{dx^2} + v_r \frac{d^3r}{dx^3} + 3v_{rrx} \left(\frac{dr}{dx}\right)^2 + v_{rrr} \left(\frac{dr}{dx}\right)^3 + 3v_{rr} \frac{dr}{dx} \frac{d^2r}{dx^2}$$

where the partial derivatives u_x, u_r, u_{xx}, u_{xr}, u_{rr}, u_{xxx}, u_{xxr}, u_{rrx}, u_{rrr}, v_x, v_r, v_{xx}, v_{xr}, v_{rr}, v_{xxx}, v_{xxr}, v_{rrx}, and v_{rrr} have been obtained by differentiation of the expressions for u and v given above.

5.6.23 Subroutine PRØP

Given the coordinate position (r,z) the gas properties u_g, v_g, T_g/T_{g_o}, and ρ_g are calculated from transonic coefficients previously determined. The velocity derivitives u_{g_z}, v_{g_z}, u_{g_r}, and v_{g_r}, may also be computed.

The quantity I in the calling sequence is a flag such that:

I = 1 implies u_g, v_g T_g/T_{g_o}, and ρ_g are to be calculated

I = 2 implies u_{g_z}, v_{g_z}, u_{g_r}, and v_{g_r}, are to be calculated

$$q_3 = \frac{k-1}{k+1}$$

$$q_2 = z_1 + x_o - z_w = z$$

$$P_o = u_o + \alpha z + \beta \frac{z^2}{2} + \gamma \frac{z^3}{6}$$

<u>r = o</u>, Properties

$$u_g = u_g^* P_o$$

$$v_g = 0$$

$$\frac{T_g}{T_{g_o}} = 1 - \frac{k-1}{k+1} \frac{u_g^2}{u_g^{*2}}$$

$$\rho_g = \rho_{g_o} \left(\frac{T_g}{T_{g_o}}\right)^{\frac{1}{k-1}}$$

<u>r ≠ o</u> Properties

$$P_2 = a_{21} + b_{21} + 2(b_{22} + c_{22})\, z + 3(c_{23})\, z^2$$

$$P_4 = b_{41} + c_{41} + 2(b_{42} + c_{42})\, z + 3c_{43} z^2$$

$$P_6 = b_{61} + c_{61} + 2c_{62} z + 3c_{63} z^2$$

$$P_8 = c_{81} + 2c_{82} z$$

$$P_{10} = c_{101}$$

$$P_1 = a_{20} + (a_{21} + b_{21})\, z + (b_{22} + c_{22})\, z^2 + c_{23} z^3$$

$$P_3 = a_{40} + b_{40} + c_{40} + (b_{41} + c_{41})\, z + (b_{42} + c_{42})\, z^2 + c_{43} z^3$$

$$P_5 = b_{60} + c_{60} + (b_{61} + c_{61})\, z + (c_{62})\, z^2 + c_{63} z^3$$

$$P_7 = b_{80} + c_{80} + c_{81} z + c_{82} z^2$$

$$P_9 = c_{100} + c_{101} z$$

$$P_{11} = c_{120}$$

$$Q_4 = r^2$$

$$u_g = u_g^* \left[P_o + P_2 r^2 + P_4 r^4 + P_6 r^6 + P_8 r^8 + P_{10} r^{10} \right]$$

$$v_g = 2\, u_g^* \left[P_1 r + 2\, P_3 r^3 + 3\, P_5 r^5 + 4\, P_7 r^7 + 5\, P_9 r^9 + 6\, P_{11} r^{11} \right]$$

$$\frac{T_g}{T_{g_o}} = 1 - \frac{k-1}{k+1} \left(\frac{u_g^2 + v_g^2}{u_g^{*2}} \right)$$

$$\rho_g = \rho_{g_o} \left(\frac{T_g}{T_{g_o}} \right)^{\frac{1}{k-1}}$$

<u>r = o</u>, derivatives

$$G_o = \alpha + \beta z + \gamma \frac{z^2}{2}$$

$$u_{g_z} = u_g^* G_o$$

$$v_{g_r} = 2\, u_g^* P_1$$

$$u_{g_r} = 0$$

$$v_{g_z} = 0$$

$r \neq o$, derivatives

$$u_{g_r} = u_g^* \left[2 P_2 r + 4 P_4 r^3 + 6 P_6 r^5 + 8 P_8 r^7 + 10 P_{10} r^9 \right]$$

$$v_{g_z} = u_{g_r}$$

$$v_{g_r} = u_g^* \left[2 P_1 + 12 P_3 r^2 + 30 P_5 r^4 + 56 P_7 r^6 + 90 P_9 r^8 + 132 P_{10} r^{10} \right]$$

$$G_2 = 2(b_{22} + c_{22}) + 6 c_{23} z$$

$$G_4 = 2(b_{42} + c_{42}) + 6 c_{43} z$$

$$G_6 = 2 c_{62} + 6 c_{63} z$$

$$G_8 = 2 c_{82}$$

$$u_{g_z} = u_g^* \left[G_o + G_2 r^2 + G_4 r^4 + G_6 r^6 + G_8 r^8 \right]$$

5.6.24 Subroutine TRACE

This subroutine integrates particle trajectories through Region II of Figure 5.6-2 using the differential equations:

$$\frac{dr_j}{dz} = \frac{v_{p_j}}{u_{p_j}}$$

$$\frac{du_{p_j}}{dz} = a_o \frac{(u_g - u_{p_j})}{u_{p_j}}$$

$$\frac{dv_{p_j}}{dz} = a_o \frac{(v_g - v_{p_j})}{u_{p_j}}$$

$$\frac{dT_{p_j}}{dz} = \frac{-b_o}{c_{p\ell}} \frac{(T_{p_j} - T_g)}{u_{p_j}}$$

where

$$a_o = \frac{9}{2} \frac{\mu_{g_o} z^*}{m_p} \left(\frac{T_g}{T_{g_o}}\right)^N \frac{f'_{p_j}}{r^2_{p_j}}$$

$$b_o = a_o \frac{2}{3} \frac{c_{pg}}{P_r} \frac{g'_{p_j}}{f'_{p_j}}$$

with f'_{p_j} and g'_{p_j} as defined in subroutine ØNED.

Initial conditions are computed by subroutine PARTIL using subroutine ØNED. The initial position ($r_{o_i} = r_o$, z_o) is selected on the upstream boundary of Region I, Figure 5.6-1.

The integration is terminated with the intersection of the particle trajectories and the end line of Region II, Figure 5.6-1. This end line is the parabola

$$z = z_{axis} - \left(\frac{z_{axis} - z_{wall}}{r^2_{wall}} \right) r^2$$

5.6.25 Subroutine WDGI

This subroutine integrates along the initial supersonic data line for mass flow (units of lbs mass) by trapezoidal rule:

$$\dot{m}_g = -\frac{\pi}{2} r^{*2} \sum_{i=1}^{50} \left[\rho_{g_{i+1}} u_{g_{i+1}} + \rho_{g_i} u_{g_i} - \frac{z_{i+1} - z_i}{r_{i+1} - r_i} \left(\rho_{g_{i+1}} v_{g_{i+1}} + \rho_{g_i} v_{g_i} \right) \right] \cdot (r_{i+1}^2 - r_i^2)$$

The initial data line is a parabola defined as

$$r_{i+1} = r_i - .02$$

$$z_{i+1} = z_{axis} - \left(\frac{z_{axis} - z_{wall}}{r_{wall}^2} \right) r_{i+1}^2$$

with

$$r_1 = 1$$

$$i = 1,2,\ldots,50$$

Particle mass flows are also computed:

$$\dot{m}_{p_j} = \left(\frac{\Sigma \dot{w}_{p_j}}{\dot{w}_g} \right) \dot{m}_g \qquad j=1,\ldots,j_{max}$$

5.6.26 Program LINK53

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to other computer systems. The only function of this routine is to call subroutine N2MAIN.

5.6.27 Subroutine ACOMP

Given two adjacent points, m and n, whose properties are known, the Reynold's number associated with the jth particle size at a third point in the neighborhood is assumed to be:

$$Re_{j_3} = \frac{1}{2}\left(Re_{j_m} + Re_{j_n}\right)$$

where

$$Re_j = \frac{2\rho_g r_{p_j}\sqrt{(u_g - u_{p_j})^2 + (v_g - v_{p_j})^2}}{\mu_{g_o}\left[\frac{P_g}{\rho_g R T_{g_o}}\right]^N}$$

A drag coefficient is found by quadratic (or linear) interpolation on Reynold's number from Table 5.6-1.

$$f_p = f_p\ (Re_j).$$

A slipflow correction is applied to the drag coefficient:

$$f'_{p_j} = f_p\left[\frac{(1+9.45c)(1+2.52c)+3.03c^2}{(1+9.45c)(1+3.78c)+.91(4+11.34c)c^2}\right]$$

$$g'_{p_j} = \frac{f_p}{1 + \left(\frac{3.42}{P_r\sqrt{\gamma}}\right) c\ f_p}$$

where

$$c = \frac{2}{Re_j}\sqrt{\frac{(u_g - u_{p_j})^2 + (v_g - v_{p_j})^2}{RT_g}}$$

The coefficient, A_{p_j}, is defined as

$$A_{p_j} = \frac{9}{2}\ \frac{\mu_{g_o} r^*}{(T_{g_o} R)^N}\ \ \frac{f'_{p_j}}{m_p r^2_{p_j}}$$

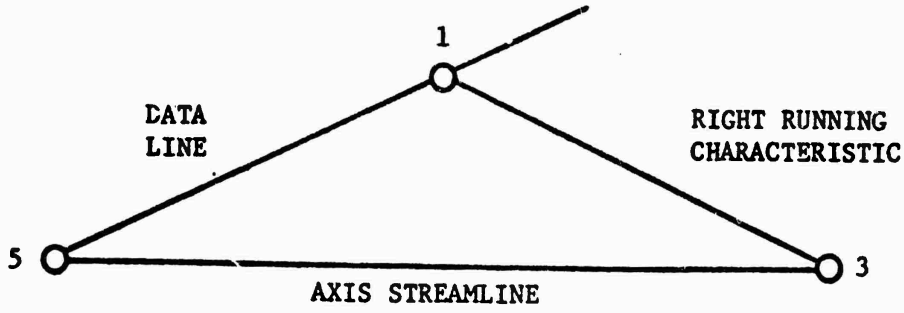
5.6.28 Subroutine ADJK

This subroutine performs Step 4 as described in subroutine KPBPT. The two arguments in its calling sequence, CALL ADJK(BARL,Q) are

BARL	$= \bar{\lambda}$	input
Q	$= q_1$	output

5.6.29 Subroutine AXISPT

This sub.outine uses two known points to calculate a new axis point



Properties at point 3 are computed from properties at points 1 and 5. The gas and particle streamlines passing through point 3 lie along the axis.

The following steps specify an axis point calculation. Tests are made to avoid particle calculations when no particles are present.

STEP 1 and STEP 2

Each variable at point 3 is set equal to the average of the corresponding variable at points 1 and 5 except:

$$r_3 = v_{g_3} = v_{p_{j_3}} = 0.$$

STEP 3

Previous values of point 3 are saved, then:

$$\bar{\lambda} = T\,A\,(\lambda_1,\ \lambda_3)$$

$$z_3 = z_1 - \frac{r_1}{\bar{\lambda}}$$

STEP 4

$$u_{p_{j_3}} = u_{p_{j_5}} + \frac{A_{p_j}}{2} (z_3 - z_5)(B_{j_3} + B_{j_5})$$

$$h_{p_{j_3}} = h_{p_{j_5}} + \frac{1}{2} (z_3 - z_5)(E_{j_3} + E_{j_5})$$

$$\rho_{p_{j_3}} = \frac{1}{u_{p_{j_3}}} \left[\rho_{p_{j_5}} u_{p_{j_5}} - \frac{2\rho_{p_{j_5}} v_{p_{j_5}}}{r_1} (z_3 - z_5) \right]$$

STEP 5

$$u_{g_3} = \frac{2(J_3 - J_1)}{L_1 - G_5 - G_3}$$

$$P_{g_3} = J_3 - \frac{L_1 u_{g_3}}{2}$$

$$\rho_{g_3} = \rho_{g_5} + \frac{(S_3+S_5)(P_{g_3} - P_{g_5}) - (\bar{T}_3+\bar{T}_5)(z_3-z_5)}{2}$$

Steps 3 through 5 are iterated until the calculated variables change by less than a prescribed amount, or until a maximum number of iterations is reached.

5.6.30 Subroutine CHECK

This subroutine compares current and previous values of the point 3 flow variables P_g, ρ_g, u_g, v_g, h_{p_j}, ρ_{p_j}, u_{p_j} and v_{p_j} for relative convergence by calling subroutine CRIT. If a variable is encountered failing the test the program returns with the convergence flag set as failed (NØ = 1). If convergence is achieved the subroutine returns with the convergence flag set as passed (NØ = 0).

5.6.31 Subroutine CNTRL

The purpose of this subroutine is to control the construction of the finite difference mesh for the method of characteristics solution of the supersonic nozzle flow. Left running characteristics are constructed starting at the nozzle throat point (r = 1, z = 0). These left running characteristics extend from the initial data line or nozzle axis to the wall. The mesh points are calculated by subroutines INPT, AXISPT, WLPT and KPBPT under control of this subroutine.

Additional points are inserted in the mesh by subroutine CNTRL by property averaging. Points are inserted to control the maximum mesh spacing. In a nozzle of shallow radius of curvature at the throat, the limiting particle streamlines may be very closely spaced. A point insertion and drop technique allows this situation to be handled without the introduction of an unnecessarily large number of characteristics. No provisions have been made for the crossing of particle trajectories or for the occurrence of shocks. Weak shocks, however, will be ignored by point deletion or characteristic mapping. Figure 5.6-31 illustrates a typical mesh construction containing Kth particle boundary points, interior points, initial line points, and points inserted and deleted.

Whenever a mesh point calculation or insertion is completed successfully, the output routine, PRINT, is called so that the point may be printed.

Subroutine CNTRL also integrates the wall pressure by trapezoidal rule to determine the axial component of force existing between adjacent wall points,

$$\Delta F = 2\pi r^{*2} \int_{r_5}^{r_2} P_g \, r\,dr = \frac{\pi}{2}(P_{g_2} + P_{g_5})(r_2^2 - r_5^2)r^{*2} .$$

Total thrust is then found by

$$F_t = F + 2\pi r^{*2} \int_{1}^{r_2} P_g \, r\,dr$$

where F is the thrust across the initial line as calculated by subroutine CØNSTS.

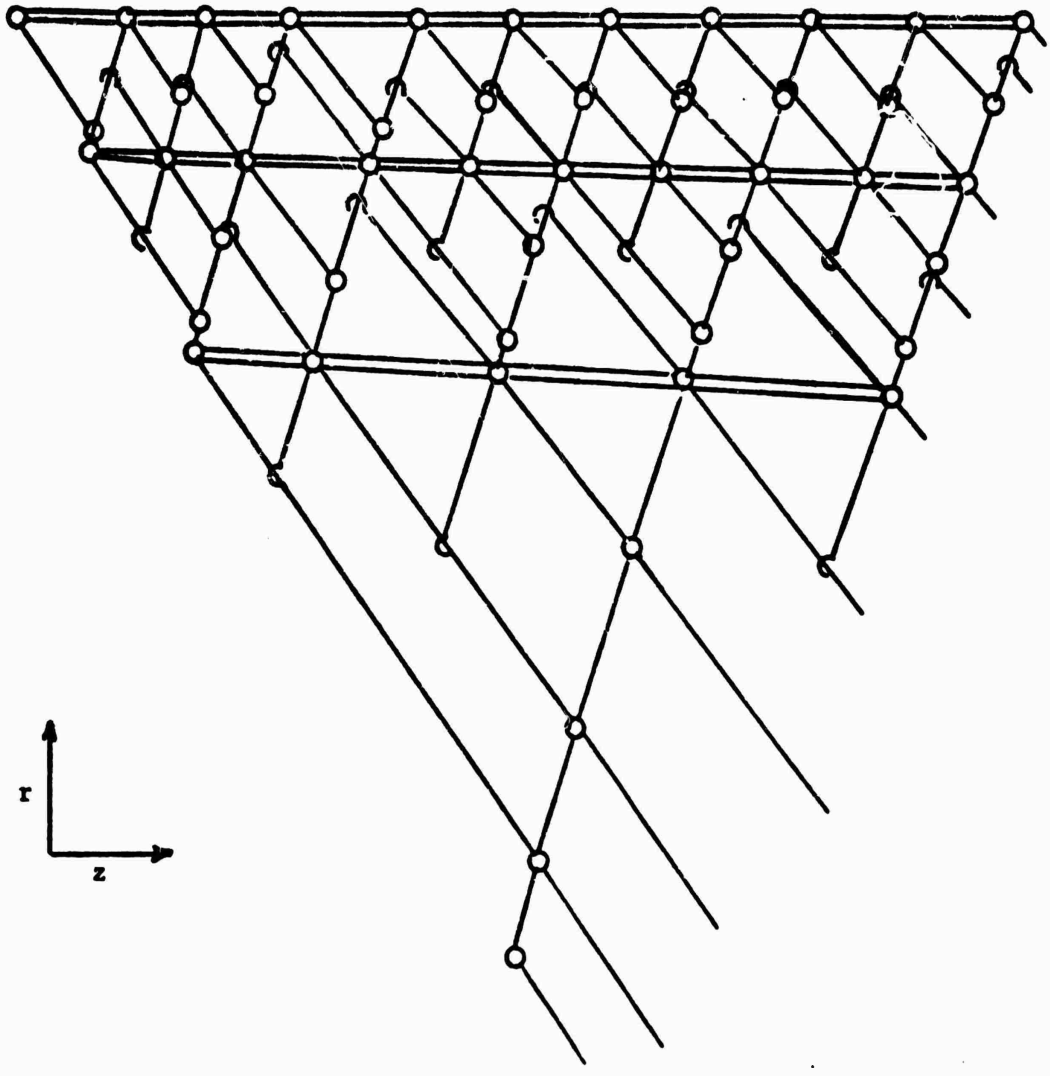


Figure 5.6-3 Typical Mesh Construction

The quantities defined below are not computed in subroutine CNTRL but are basic to the point calculations controlled by subroutine CNTRL. These quantities are given here for convenience.

$$1^R = \sqrt{\gamma P_g}$$

$$2^R = \sqrt{\rho_g (u_g^2 + v_g^2) - \gamma P_g}$$

$$3^R = \frac{1^R}{2^R}$$

$$1^V = u_g^2 \rho_g - \gamma P_g$$

$$\kappa = \frac{\rho_g u_g v_g + 1^R 2^R}{1^V}$$

$$\lambda = \frac{\rho_g u_g v_g - 1^R 2^R}{1^V}$$

$$C_g = \frac{v_g}{u_g}$$

$$C_{p_j} = \frac{v_{p_j}}{u_{p_j}}$$

$$B_{p_j} = \left(\frac{P_g}{\rho_g}\right)^N \frac{(u_g - u_{p_j})}{u_{p_j}}$$

$$D_{p_j} = \left(\frac{P_g}{\rho_g}\right)^N \frac{(v_g - v_{p_j})}{u_{p_j}}$$

$$F = 3^R \left(\frac{P_g}{\rho_g}\right)^N [{}_4t - \frac{\gamma P_g v_g}{r}]$$

$$\mu = \frac{\rho_g (v_g 1^R + u_g 2^R)}{1^R 1^V}$$

$$\eta = \frac{\rho_g (v_g 1^R - u_g 2^R)}{1^R 1^V}$$

$$G = \rho_g u_g + \frac{z_3 - z_5}{2}\left(\frac{P_g}{\rho_g}\right)^N {}_1t$$

$$H = \rho_g v_g + \frac{r_3 - r_5}{2}\left(\frac{P_g}{\rho_g}\right)^N {}_1t$$

$$L = \frac{1^R}{2^R}\left[\rho_g v_g + \frac{r_3}{2}\left(\frac{P_g}{\rho_g}\right)^N {}_1t\right]$$

$$Q = -\frac{1^R}{2^R}\left[\rho_g u_g + \frac{z_3}{2}\left(\frac{P_g}{\rho_g}\right)^N {}_1t\right]$$

$$S = \frac{\rho_g}{\gamma P_g}$$

$$\bar{T} = \frac{S}{u_g}\left(\frac{P_g}{\rho_g}\right)^N {}_4t$$

$$J_1 = P_{g_5} + \frac{r_3 - r_5}{2}\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_3t_3 + \left(\frac{P_{g_5}}{\rho_{g_5}}\right)^N {}_3t_5\right]$$

$$+ \frac{z_3 - z_5}{2}\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_2t_3 + \left(\frac{P_{g_5}}{\rho_{g_5}}\right)^N {}_2t_5\right]$$

$$- \frac{u_{g_5}(z_3 - z_5)}{4}\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_1t_3 + \left(\frac{P_{g_5}}{\rho_{g_5}}\right)^N {}_1t_5\right]$$

$$- \frac{v_{g_5}(r_3 - r_5)}{4}\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_1t_3 + \left(\frac{P_{g_5}}{\rho_{g_5}}\right)^N {}_1t_5\right]$$

$$+ \frac{v_{g_5}}{2}\left[\rho_{g_3} v_{g_3} + \rho_{g_5} v_{g_5}\right] + \frac{u_{g_5}}{2}\left[\rho_{g_3} u_{g_3} + \rho_{g_5} u_{g_5}\right]$$

$$J_2 = P_{g_2} + \frac{z_3 - z_2}{2}(F_3 \mu_3 + F_2 \mu_2) + \frac{1}{2}({}_3R_2 + {}_3R_3)$$

$$\left\{ - \frac{u_{g_2}}{2}(\rho_{g_2} v_{g_2} + \rho_{g_3} v_{g_3}) + \frac{v_{g_2}}{2}(\rho_{g_2} u_{g_2} + \rho_{g_3} u_{g_3}) \right.$$

$$- \frac{(r_3 - r_2)}{2}\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_2t_3 + \left(\frac{P_{g_2}}{\rho_{g_2}}\right)^N {}_2t_2\right]$$

$$+ \frac{(z_3 - z_2)}{2}\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_3t_3 + \left(\frac{P_{g_2}}{\rho_{g_2}}\right)^N {}_3t_2\right]$$

$$+ \frac{1}{4}\left[u_{g_2}(r_3 - r_2) - v_{g_2}(z_3 - z_2) - u_{g_3} r_2 + v_{g_3} z_2\right]$$

$$\left.\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_1t_3 + \left(\frac{P_{g_2}}{\rho_{g_2}}\right)^N {}_1t_2\right]\right\}$$

$$J_3 = P_{g_1} - \frac{(z_3 - z_1)}{2}(F_3 \eta_3 + F_1 \eta_1) - \frac{1}{2}({}_3R_1 + {}_3R_3)$$

$$\left\{ - \frac{u_{g_1}}{2}(\rho_{g_1} v_{g_1} + \rho_{g_3} v_{g_3}) + \frac{v_{g_1}}{2}(\rho_{g_1} u_{g_1} + \rho_{g_3} u_{g_3}) \right.$$

$$- \frac{(r_3 - r_1)}{2}\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_2t_3 + \left(\frac{P_{g_1}}{\rho_{g_1}}\right)^N {}_2t_1\right]$$

$$+ \frac{(z_3 - z_1)}{2}\left[\left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_3t_3 + \left(\frac{P_{g_1}}{\rho_{g_1}}\right)^N {}_3t_1\right]$$

$$+ \frac{1}{4}\left[u_{g_1}(r_3 - r_1) - v_{g_1}(z_3 - z_1) - u_{g_3} r_1 + v_{g_3} z_1\right]$$

$$\left.\left[\left(\frac{P_{g_1}}{\rho_{g_1}}\right)^N {}_1t_1 + \left(\frac{P_{g_3}}{\rho_{g_3}}\right)^N {}_1t_3\right]\right\}$$

Note: When calculating an axis point the evaluation of F_3 requires an indeterminate ($v_g = r = 0$) quantity, $\frac{v_g}{r}$. In this case the expression below is used:

$$\left(\frac{v_g}{r}\right)_3 = \frac{v_{g_1}}{r_1}\left[\frac{u_{g_3}}{u_{g_1} + (z_3 - z_1)\frac{v_{g_1}}{r_1}}\right]$$

This calculation amounts to extrapolating for the ratio, $\frac{v_g}{r}$, assuming that the flow near the axis is a source flow.

5.6.32 Subroutine CONSTS

An integration is performed using trapezoidal rule for thrust along the initial line:

$$F = 2\pi r^{*2} \int_0^{r_{wall}} \left[P_g + \rho_g u_g (u_g - v_g) + \sum_{j=1}^{K} \rho_{p_j} u_{p_j} (u_{p_j} - v_{p_j}) \right] r dr,$$

where K is the total number of particle sizes present at the point under consideration.

The indeterminate quantity

$$\frac{v_g}{r}$$

at the initial line axis point, m, is calculated as

$$\left(\frac{v_g}{r}\right)_j = \frac{v_{g_{m-1}}}{r_{m-1}} \left[\frac{u_{g_m}}{u_{g_{m-1}} + (z_m - z_{m-1}) \frac{v_{g_{m-1}}}{r_{m-1}}} \right]$$

This quantity is required in both the axis and off axis point calculations.

5.6.33 Subroutine CRIT

The purpose of this subroutine is to compare two values for relative convergence and return a flag stating if convergence has been achieved.

Calling Sequence:

XN x_n

XM x_m

NO A flag such that

if $x_n = 0$ then NO = 0

if $|(x_m - x_n)/x_n| \leq 5.10^{-5}$ then NO = 0

if $|(x_m - x_n)/x_n| > 5.10^{-5}$ then NO = 1

5.6.34 Subroutine CUBIC (X, Y, YP, N, ARG, YARG)

The purpose of this subroutine is to perform cubic interpolation for a tabulated function whose derivatives are known.

Calling Sequence:

X is a table of the independent variable, x_i, such that $x_{i+1} \geq x_i$

Y is a table of the dependent variable, $y_i = y(x_i)$

YP is a table of the derivitives of the dependent variable, $y_i' = y'(x_i)$

N is the number of entries in each of the above tables; $i = 1, \cdots N$

ARG is the argument, x, for which interpolation is requested

YARG is the result, $y = y(x)$

Restrictions:

The calling program must define arrays for the dummy variables X, Y, and YP. These arrays must be at least of lengths N + 1, N, and N respectively. The subroutine will save its last used table position number in X(N + 1).

If $x < x_1$ the program returns $y = y_1$

If $x > x_N$ the program returns $y = y_N$

Method:

Given: $x_o \leq x < x_1$

so that y_o, y_o', y_1, and y_1' are known. The cubic interpolation formula given below is used to determine y

$$y = A(x - x_o)^3 + B(x - x_o)^2 + C(x - x_o) + D$$

where

$$A = \frac{1}{h^3}\left[\left(y_1' + y_o'\right) h - 2k\right]$$

$$B = \frac{-1}{h^2}\left[\left(y_1' + 2y_o'\right) h - 3k\right]$$

$$C = y_o'$$

$$D = y_o$$

and

$$h = x_1 - x_o$$

$$k = y_1 - y_o$$

5.6.35 Function EFN

The quantity E_j is calculated in this function subroutine. A test is made on particle enthalpy, h_{p_j}, in order to determine the proper particle phase.

$$E_j = \frac{-b_{p_j}\left(\frac{P_g}{\rho_g}\right)^N g^1}{u_{p_j}} \left[T_{p_m} + \frac{h_{p_j} - h_{p_\ell}}{C_{p_\ell}} - \frac{P_g}{R\,\rho_g} \right] \qquad h_{p_j} > h_{p_\ell}$$

$$E_j = \frac{-b_{p_j}\left(\frac{P_g}{\rho_g}\right)^N g^1}{u_{p_j}} \left[T_{p_m} - \frac{P_g}{R\,\rho_g} \right] \qquad h_{p_s} \leq h_{p_j} \leq h_{p_\ell}$$

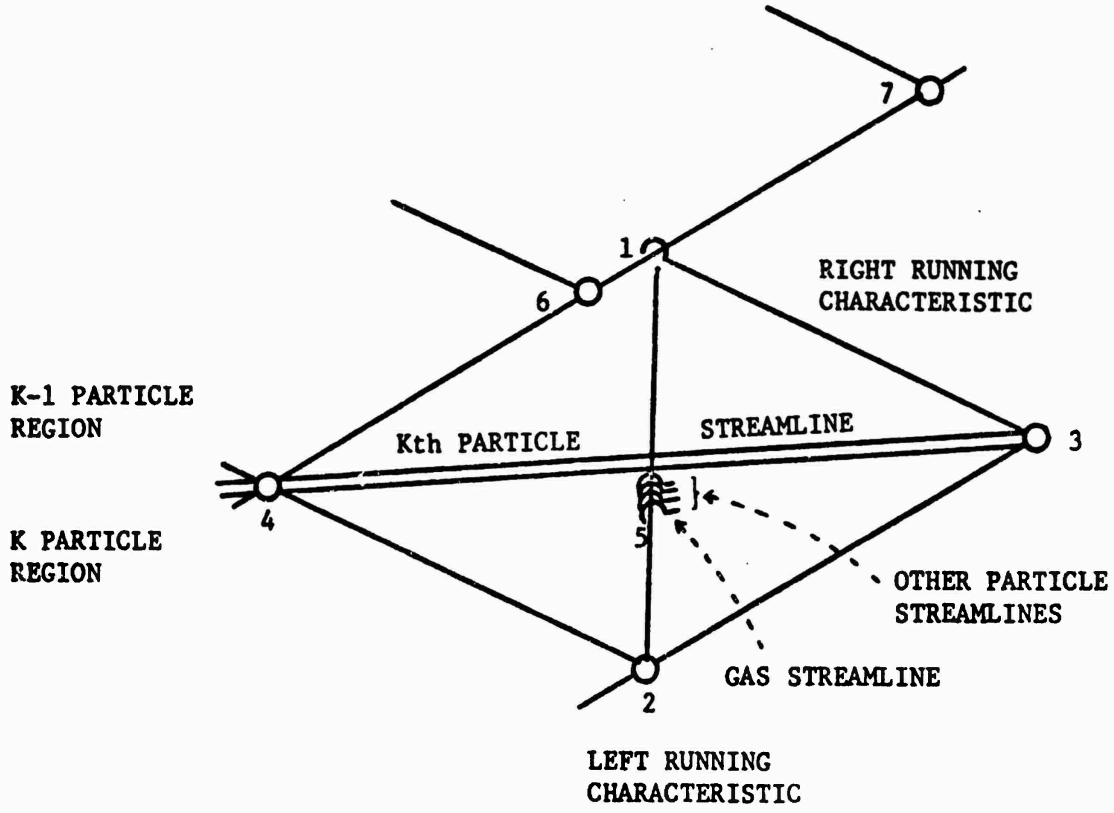
$$E_j = \frac{-b_{p_j}\left(\frac{P_g}{\rho_g}\right)^N g^1}{u_{p_j}} \left[\frac{h_{p_s} - h_{p_j}}{C_{p_s}} - \frac{P_g}{R\,\rho_g} \right] \qquad h_{p_j} < h_{p_s}$$

5.6.38 Subroutine ERRØR

The purpose of this subroutine is to print error messages as requested by subroutines CNTRL, ADJK, and WLPT. See Appendix A for conversion aides.

5.6.37 Subroutine KPBPT

This subroutine uses five known points to calculate a Kth particle boundary.



Properties at point 3 are computed from properties at point 2 and interpolated properties at point 1.

The following steps specify a Kth particle boundary point calculation. Tests are made to simplify the calculations when a gas and one particle boundary point is to be computed.

Step 1

Each variable at point 3 is set equal to the average of the corresponding variable at points 2 and 4. Point 1 is initially set equal to point 4.

Then

$$u_{p_{K_1}} = u_{p_{K_2}}$$

$$v_{p_{K_1}} = v_{p_{K_2}}$$

$$u_{g_5} = u_{g_3}$$

$$v_{g_5} = v_{g_3}$$

$$u_{p_{j_4}} = u_{p_{j_3}} \qquad j = 1, \ldots, K-1$$

$$v_{p_{j_4}} = v_{p_{j_3}} \qquad j = 1, \ldots, K-1$$

Step 2

Previous values of point 3 are saved, then

$$\bar{\kappa} = TA(\kappa_2, \kappa_3)$$

$$\bar{C}_{p_K} = TA(C_{p_{K_4}}, C_{p_{K_3}})$$

$$z_3 = \frac{r_4 - r_2 + \bar{\kappa} z_2 - \bar{C}_{p_K} z_4}{\bar{\kappa} - \bar{C}_{p_K}}$$

$$r_3 = r_2 + \bar{\kappa}(z_3 - z_2)$$

Step 3

$$\bar{\lambda} = TA(\lambda_3, \lambda_1)$$

Points 6 and 7 are selected such that point 6 is point 4 and point 7 is the next point on the previous characteristic.

Step 4

This step is performed by subroutine ADJK.

$$\kappa^* = \frac{r_7 - r_6}{z_7 - z_6}$$

$$z_1 = \frac{r_3 - r_6 + \kappa^* z_6 - \bar{\lambda} z_3}{\kappa^* - \bar{\lambda}}$$

$$q_1 = \frac{z_1 - z_6}{z_7 - z_6}$$

If $q_1 > 1$ select new points 6 and 7 which are one point higher on the previous left running characteristic and restart Step 4.

If $q_1 < 0$ select new points 6 and 7 which are one point lower on the previous left running characteristic and restart Step 4.

If $1 > q_1 > 0$ proceed.

Step 5

$$r_1 = r_6 + q_1(r_7 - r_6)$$

$$P_{g_1} = P_{g_6} + q_1(P_{g_7} - P_{g_6})$$

$$\rho_{g_1} = \rho_{g_6} + q_1(\rho_{g_7} - \rho_{g_6})$$

$$u_{g_1} = u_{g_6} + q_1(u_{g_7} - u_{g_6})$$

$$v_{g_1} = v_{g_6} + q_1(v_{g_7} - v_{g_6})$$

$$h_{p_{j_1}} = h_{p_{j_6}} + q_1(h_{p_{j_7}} - h_{p_{j_6}}) \qquad j = 1, \ldots, K-1$$

$$\rho_{p_{j_1}} = \rho_{p_{j_6}} + q_1(\rho_{p_{j_7}} - \rho_{p_{j_6}}) \qquad j = 1, \ldots, K-1$$

$$u_{p_{j_1}} = u_{p_{j_6}} + q_1(u_{p_{j_7}} - u_{p_{j_6}}) \qquad j = 1, \ldots, K-1$$

$$v_{p_{j_1}} = v_{p_{j_6}} + q_1(v_{p_{j_7}} - v_{p_{j_6}}) \qquad j = 1, \ldots, K-1$$

Step 6

Steps 6 and 7 are performed for each j = 1, ..., K. The gas properties at point 4 are first saved.

If j = K — restore gas properties at point 4 and go to Step 7.

If j ≠ K — proceed.

$$\bar{C}_{p_j} = TA(C_{p_{j_3}}, C_{p_{j_4}})$$

$$r_4 = \frac{r_3 + \bar{C}_{p_j}\left(z_2 - z_3 - r_2 \frac{z_1 - z_2}{r_1 - r_2}\right)}{1 - \bar{C}_{p_j} \frac{z_1 - z_2}{r_1 - r_2}}$$

$$q_4 = \frac{r_4 - r_2}{r_1 - r_2}$$

$$z_4 = z_2 + q_4(z_1 - z_2)$$

$$P_{g_4} = P_{g_2} + q_4(P_{g_1} - P_{g_2})$$

$$\rho_{g_4} = \rho_{g_2} + q_4(\rho_{g_1} - \rho_{q_2})$$

$$u_{g_4} = u_{g_2} + q_4(u_{g_1} - u_{g_2})$$

$$v_{g_4} = v_{g_2} + q_4(v_{g_1} - v_{g_2})$$

$$h_{p_{j_4}} = h_{p_{j_2}} + q_4(h_{p_{j_1}} - h_{p_{j_2}})$$

$$\rho_{p_{j_4}} = \rho_{p_{j_2}} + q_4(\rho_{p_{j_1}} - \rho_{p_{j_2}})$$

$$u_{p_{j_4}} = u_{p_{j_2}} + q_4(u_{p_{j_1}} - u_{p_{j_2}})$$

$$v_{p_{j_4}} = v_{p_{j_2}} + q_4(v_{p_{j_1}} - v_{p_{j_2}})$$

Step 7

$$u_{p_{j_3}} = u_{p_{j_4}} + \frac{A_j}{2}(z_3 - z_4)(B_{j_3} + B_{j_4})$$

$$v_{p_{j_3}} = v_{p_{j_4}} + \frac{A_j}{2}(z_3 - z_4)(D_{j_3} + D_{j_4})$$

$$h_{p_{j_3}} = h_{p_{j_4}} + \frac{1}{2}(z_3 - z_4)(E_{j_3} + E_{j_4})$$

Step 8

for $j = K$

$$\rho_{p_{j_3}} = \frac{1}{[u_{p_{j_3}}(r_3^2 - r_2^2) - v_{p_{j_3}}(z_3 - z_3)(r_2 + r_2)]} \Big\{ \rho_{p_{j_4}} [u_{p_{j_4}}(r_4^2 - r_2^2)$$

$$- v_{p_{j_4}}(z_4 - z_2)(r_2 + r_4)] + \rho_{p_{j_2}} [u_{p_{j_2}}(r_4^2 - r_3^2)$$

$$- v_{p_{j_2}} [(z_4 - z_2)(r_4 + r_2) - (z_3 - z_2)(r_2 + r_3)]] \Big\}$$

for $j < K$

$$\rho_{p_{j_3}} = \frac{1}{u_{p_{j_3}}(r_1^2 - r_2^2) - v_{p_{j_3}}[(z_3 - z_2)(r_2 + r_3) + (z_1 - z_3)(r_1 + r_3)]} \Big\{$$

$$\rho_{p_{j_4}} [u_{p_{j_4}}(r_1^2 - r_2^2) - v_{p_{j_4}}[(z_4 - z_2)(r_4 + r_2) + (z_1 - z_4)(r_4 + r_1)]]$$

$$+ \rho_{p_{j_1}} [u_{p_{j_1}} (r_3^2 - r_4^2) - v_{p_{j_1}} [(z_1 - z_4)(r_4 + r_1) - (z_1 - z_3)(r_1 + r_3)]]$$

$$+ \rho_{p_{j_2}} [u_{p_{j_2}} (r_4^2 - r_3^2) - v_{p_{j_2}} [(z_4 - z_2)(r_4 + r_2) - (z_3 - z_2)(r_3 + r_2)]]\}$$

Step 9

$$\bar{C}_g = TA(C_{g_3}, C_{g_5})$$

$$r_5 = \frac{r_3 + \bar{C}_g[z_2 - z_3 - r_2 \frac{z_1 - z_2}{r_1 - r_2}]}{1 - \bar{C}_g \frac{z_1 - z_2}{r_1 - r_2}}$$

$$q_5 = \frac{r_5 - r_2}{r_1 - r_2}$$

$$z_5 = z_2 + q_5(z_1 - z_2)$$

$$P_{g_5} = P_{q_2} + q_5(P_{g_1} - P_{g_2})$$

$$\rho_{g_5} = \rho_{g_2} + q_5(\rho_{g_1} - \rho_{g_2})$$

$$u_{g_5} = u_{g_2} + q_5(u_{g_1} - u_{g_2})$$

$$v_{g_5} = v_{g_2} + q_5(v_{g_1} - v_{g_2})$$

$$h_{p_{j_5}} = h_{p_{j_2}} + q_5(h_{p_{j_1}} - h_{p_{j_2}}) \qquad j = 1, \ldots, K-1$$

$$\rho_{p_{j_5}} = \rho_{p_{j_2}} + q_5(\rho_{p_{j_1}} - \rho_{p_{j_2}}) \qquad j = 1, \ldots, K-1$$

$$u_{p_{j_5}} = u_{p_{j_2}} + q_5(u_{p_{j_1}} - u_{p_{j_2}}) \qquad j = 1, \ldots, K-1$$

$$v_{p_{j_5}} = v_{p_{j_2}} + q_5(v_{p_{j_1}} - v_{p_{j_2}}) \qquad j = 1, \ldots, K-1$$

A test is made to determine whether point 5 is above or below the Xth particle streamline.

If

$$\frac{r_3 - r_4}{z_3 - z_4} > \frac{r_3 - r_5}{z_3 - z_5}$$

then

$h_{p_{K_5}} = \rho_{p_{K_5}} = u_{p_{K_5}} = 0$ and primes must be added in Step 10 to J_1, G_3, G_5, H_3, H_5, T_3, and T_5;

otherwise:

$$h_{p_{K_5}} = \frac{1}{2}(h_{p_{K_4}} + h_{p_{K_2}})$$

$$\rho_{p_{K_5}} = \frac{1}{2}(\rho_{p_{K_4}} + \rho_{p_{K_2}})$$

$$u_{p_{K_5}} = \frac{1}{2}(u_{p_{K_4}} + u_{p_{K_2}})$$

$$v_{p_{K_5}} = \frac{1}{2}(v_{p_{K_4}} + v_{p_{K_2}})$$

Step 10

For the primed quantities the implied summations are to be taken for $j = 1, \ldots, K-1$, not K.

$$u_{g_3} = \frac{2(J_3' - J_2)(Q_3 + Q_2 + H_5 + H_3) - 2(J_1 - J_2)(Q_3' + Q_3 + Q_1' + Q_2)}{(L_3' + L_3 + L_1' + L_2)(Q_3 + Q_2 + H_5 + H_3) - (L_3 + L_2 + G_5 + G_3)(Q_3' + Q_3 + Q_1' + Q_2)}$$

$$v_{g_3} = \frac{2(J_3' - J_2) - u_{g_3}(L_3' + L_3 + L_1' + L_2)}{Q_3' + Q_3 + Q_1' + Q_2}$$

$$P_{g_3} = J_2 + \frac{u_{g_3}(L_3 + L_2) + v_{g_3}(Q_3 + Q_2)}{2}$$

$$\rho_{g_3} = \rho_{g_5} + \frac{(S_3 + S_5)(P_{g_3} - P_{g_5}) - (\bar{T}_3 + \bar{T}_5)(z_3 - z_5)}{2}$$

Steps 2 through 10 are iterated until the calculated variables change by less than a prescribed amount or until a maximum number of iterations is reached.

5.6.38 Subroutine N2MAIN

The purpose of this subroutine is to call the two portions of the axisymmetric supersonic method of characteristics subprogram (subroutines N3MAIN and CNTRL).

5.6.39 Subroutine N3MAIN

The purpose of this subroutine is to call subroutines CØNSTS and WALL.

5.6.40 Subroutine PRINT

The purpose of this subroutine is to print output for the supersonic flow calculation as requested by the program input. Printout may occur after the completion of each mesh point calculation. Points for print are selected as follows:

1) The following points are always printed:

   axis points
   Kth particle boundary points
   initial line points
   wall points.

2) Interior points are selected for print only along every n_1th left running characteristic and only at every n_2th position along these characteristics.

3) Inserted points are printed if all points are to be printed ($n_1 = n_2 = 1$).

The items printed are listed in the table below in the order they appear, left to right, on the output sheet. A header is printed for identification purposes above each characteristic.

Row One:

Item	Header	Meaning	Units
L.R.C. number	LRC	left running characteristic number	none
Ident. number	ID	type of point (see below)	none
r	R	r position coordinate	none
z	Z	z position coordinate	none
M	MACH	Mach number	none
T_g	TG	gas temperature	^oR
V_g	VG	gas velocity (scalar)	ft/sec

Row One: (Cont'd)

Item	Header	Meaning	Units
θ_g	THETA-G	streamline angle	degrees
T_g/T_{g_o}	TG/TGO	ratio of gas temperature to chamber temperature	none
P_g/P_{g_o}	PG/PGO	ratio of gas pressure to chamber pressure	none
ρ_g/ρ_{g_o}	DG/DGO	ratio of gas density to chamber density	none
$\sum^{K} \rho_{p_j}/\rho_g$	SDK/DG	ratio of total particle density to gas density	none
C_F	CF	thrust coefficient	none
I_{sp}	ISP	specific impulse	sec.
iteration no.	IT	number of iterations required	none

Rows Two through K+1

A row is printed for each particle size, k=1,...,K.

Item	Header	Meaning	Units
k	K	particle size number	none
Re_k	REK	particle Reynolds number	none
V_{p_k}	VPK	particle velocity (scalar)	ft/sec
θ_{p_k}	THETA-K	particle streamline angle	degrees
T_{p_k}	TPK	particle temperature	^oR
ρ_{p_k}/ρ_g	DPK/DG	ratio of particle density to gas density	none
ρ_{p_k}/ρ_{p_o}	DPK/DPO	ratio of particle density to chamber particle density	none
r_{p_k}	RPK	particle radius	ft.

Of the quantities above, the following are calculated within subroutine PRINT:

$$T_g = \frac{P_g}{\rho_g R}$$

$$V_g = \sqrt{u_g^2 + v_g^2}$$

$$M = \frac{V_g}{\sqrt{\gamma k T_g}}$$

$$\theta_g = \arctan\left(\frac{v_g}{u_g}\right) \cdot 57.29578$$

$$Re_k = \frac{2 \rho_g r_{p_k}}{\mu_{g_o}} \left(\frac{\rho_g T_{g_o} R}{P_g}\right)^N \sqrt{(u_g - u_{p_k})^2 + (v_g - v_{p_k})^2} \qquad k=1,\ldots K$$

$$C_f = \frac{F_t}{\pi r_*^2 P_{g_o}}$$

$$I_{SP} = \frac{F_t}{\dot{w}_g + \sum_{k=1}^{K} \dot{w}_{p_k}}$$

$$T_{p_k} = T_{p_m} + \frac{h_{p_k} - h_{p_\ell}}{C_{p_\ell}} \qquad \text{, for } h_{p_k} > h_{p_\ell}$$

$$T_{p_k} = T_{p_m} \qquad \text{, for } h_{p_s} \leq h_{p_k} \leq h_{p_\ell}$$

$$T_{p_k} = \frac{h_{p_k}}{C_{p_s}} \qquad \text{, for } h_{p_k} < h_{p_s}$$

The table below relates the printed identification number to the type of point calculated:

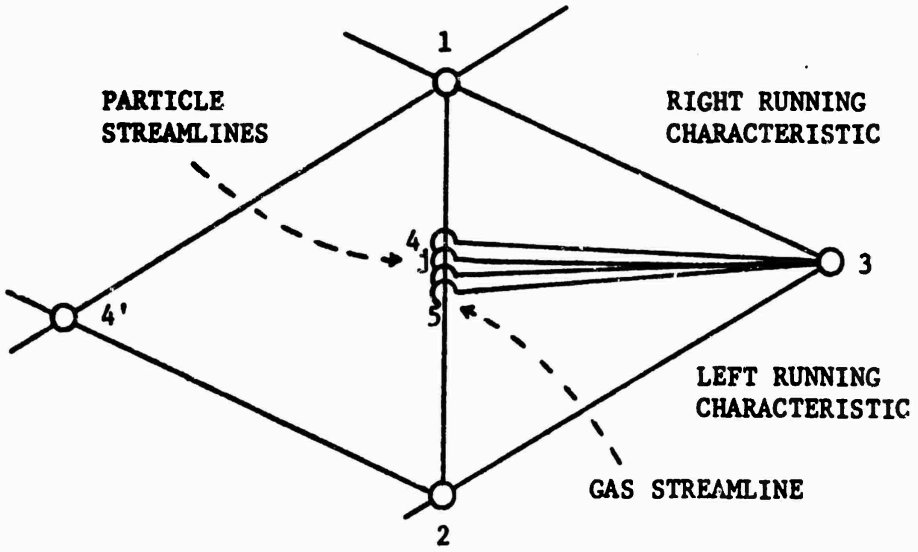
ID	Type of Point Calculated
1	initial line point
2	interior point
3	axis point
4	Kth particle boundary point
5	wall point
6	point inserted on the previous L.R.C.
7	point inserted on the R.R.C.

In addition, whenever a wall point is printed, subroutine ISP1D is called to calculate the 1D reference ISP and the quantity, η'_{TD2P} is calculated as

$$\eta'_{TD2P} = \frac{I_{sp_{TD2P}}(r_w, Z)}{I_{sp_{1DO_\ell}}(r_w^2)}$$

5.6.41 Subroutine PTINT

This subroutine uses three known points to calculate an interior point.



Properties at point 3 are computed from properties at points 1, 2, and 4. Point 4' is required only for the calculation of particle density, $\rho_{p_{j_3}}$. If point 4' is not available (IFG = 0) an alternate calculation is used (see STEP 6).

The following steps specify an interior point calculation. Tests are made to avoid particle calculations when no particles are present.

STEP 1

Each variable at point 3 is set equal to the average of the corresponding variable at points 1 and 2. Other values required for the first iterations are set as:

$$u_{g_5} = u_{g_3}$$

$$v_{g_5} = v_{g_3}$$

$$u_{p_{j_4}} = u_{p_{j_3}} \qquad j=1,\ldots,K$$

$$v_{p_{j_4}} = v_{p_{j_3}} \qquad j=1,\ldots,K$$

STEP 2

Previous values of point 3 are saved, then:

$$\bar{\kappa} = T\,A\,(\kappa_2,\ \kappa_3)$$

$$\bar{\lambda} = T\,A\,(\lambda_1,\ \lambda_3)$$

$$z_3 = \frac{r_1 - r_2 + \bar{\kappa}\, z_2 - \bar{\lambda}\, z_1}{\bar{\kappa} - \bar{\lambda}}$$

$$r_3 = r_2 + \bar{\kappa}\,(z_3 - z_2)$$

STEP 3

$$\bar{C}_g = T\,A\,(C_{g3},\ C_{g5})$$

$$r_5 = \frac{r_3 + \bar{C}_g \left[z_2 - z_3 - r_2 \frac{z_1 - z_2}{r_1 - r_2} \right]}{1 - \bar{C}_g \frac{z_1 - z_2}{r_1 - r_2}}$$

$$q_5 = \frac{r_5 - r_2}{r_1 - r_2}$$

$$z_5 = z_2 + q_5\,(z_1 - z_2)$$

$$P_{g_5} = P_{g_2} + q_5\,(P_{g_1} - P_{g_2})$$

$$\rho_{g_5} = \rho_{g_2} + q_5\,(\rho_{g_1} - \rho_{g_2})$$

$$u_{g_5} = u_{g_2} + q_5(u_{g_1} - u_{g_2})$$

$$v_{g_5} = v_{g_2} + q_5(v_{g_1} - v_{g_2})$$

$$h_{p_{j_5}} = h_{p_{j_2}} + q_5(h_{p_{j_1}} - h_{p_{j_2}}) \qquad j=1,\ldots,K$$

$$\rho_{p_{j_5}} = \rho_{p_{j_2}} + q_5(\rho_{p_{j_2}} - \rho_{p_{j_2}}) \qquad j=1,\ldots,K$$

$$u_{p_{j_5}} = u_{p_{j_2}} + q_5(u_{p_{j_1}} - u_{p_{j_2}}) \qquad j=1,\ldots,K$$

$$v_{p_{j_5}} = v_{p_{j_2}} + q_5(v_{p_{j_1}} - v_{p_{j_2}}) \qquad j=1,\ldots,K$$

STEP 4

Steps 4, 5, and 6 are repeated for each j=1,...,K.

$$\overline{C}_{p_j} = T\,A\,(C_{p_{j_3}}, C_{p_{j_4}})$$

$$r_4 = \frac{r_3 + \overline{C}_{p_j}\left[z_2 - z_3 - r_2\,\frac{z_1 - z_2}{r_1 - r_2}\right]}{1 - \overline{C}_{p_j}\,\frac{z_1 - z_2}{r_1 - r_2}}$$

$$q_4 = \frac{r_4 - r_2}{r_1 - r_2}$$

$$z_4 = z_2 + q_4(z_1 - z_2)$$

$$P_{g_4} = P_{g_2} + q_4(P_{g_1} - P_{g_2})$$

$$\rho_{g_4} = \rho_{g_2} + q_4(\rho_{g_1} - \rho_{g_2})$$

$$u_{g_4} = u_{g_2} + q_4(u_{g_1} - u_{g_2})$$

$$v_{g_4} = v_{g_2} + q_4(v_{g_1} - v_{g_2})$$

$$h_{p_{j_4}} = h_{p_{j_2}} + q_4(h_{p_{j_1}} - h_{p_{j_2}}) \qquad j=1,\ldots,K$$

$$\rho_{p_{j_4}} = \rho_{p_{j_2}} + q_4(\rho_{p_{j_1}} - \rho_{p_{j_2}}) \qquad j=1,\ldots,K$$

$$u_{p_{j_4}} = u_{p_{j_2}} + q_4(u_{p_{j_1}} - u_{p_{j_2}}) \qquad j=1,\ldots,K$$

$$v_{p_{j_4}} = v_{p_{j_2}} + q_4(v_{p_{j_1}} - v_{p_{j_2}}) \qquad j=1,\ldots,K$$

STEP 3

Calculate

$$u_{p_{j_3}} = u_{p_{j_4}} + \frac{A_{p_j}}{2}(z_3 - z_4)\,(B_{p_{j_3}} + B_{p_{j_4}})$$

$$v_{p_{j_3}} = v_{p_{j_4}} + \frac{A_{p_j}}{2}(z_3 - z_4)(D_{p_{j_3}} + D_{p_{j_4}})$$

$$h_{p_{j_3}} = h_{p_{j_4}} + \frac{1}{2}(z_3 - z_4)(E_{p_{j_3}} + E_{p_{j_4}})$$

STEP 6

If point 4 ' is not available (IFG = 0):

$$\rho_{p_{j_3}} = \frac{1}{u_{p_{j_3}}(r_1^2 - r_2^2) - v_{p_{j_3}}\left[\frac{r_3^2 - r_2^2}{\kappa_3} + (r_1 + r_3)(z_1 - z_3)\right]} \Bigg\{ \rho_{p_{j_2}}\left[u_{p_{j_2}}(r_1^2 - r_3^2) - v_{p_{j_2}}\left[(z_1 - z_2)(r_1 + r_2) - \frac{r_3^2 - r_2^2}{\kappa_2}\right]\right] + \rho_{p_{j_1}}\left[u_{p_{j_1}}(r_3^2 - r_2^2) - v_{p_{j_1}}\left[(z_1 - z_2)(r_1 + r_2) - (r_1 + r_2)(z_1 - z_3)\right]\right]\Bigg\}$$

If point 4 ' is available (IFG>0):

$$\rho_{p_{j_3}} = \frac{1}{u_{p_{j_3}}(r_1^2 - r_2^2) - v_{p_{j_3}}\left[(z_3 - z_2)(r_2 + r_3) + (z_1 - z_3)(r_1 + r_3)\right]} \Bigg\{ \rho_{p_{j_4'}}\left[u_{p_{j_4'}}(r_1^2 - r_2^2) - v_{p_{j_4'}}\left[(z_4' - z_2)(r_4' + r_2) + (z_1 - z_4')(r_4' + r_1)\right]\right] + \rho_{p_{j_1}}\left[u_{p_{j_1}}(r_3^2 - r_4'^2) - v_{p_{j_1}}\left[(z_1 - z_4')(r_4' + r_1) - (z_1 - z_3)(r_1 + r_3)\right]\right] + \rho_{p_{j_2}}\left[u_{p_{j_2}}(r_4'^2 - r_3^2) - v_{p_{j_2}}\left[(z_4' - z_2)(r_4' + r_2) - (z_3 - z_2)(r_3 + r_2)\right]\right]\Bigg\}$$

STEP 7

$$u_{g_3} = \frac{2(J_3 - J_2)(Q_3 + Q_2 + H_5 H_3) - 2(J_1 - J_2)(2Q_3 + Q_1 + Q_2)}{(2L_3 + L_1 + L_2)(Q_3 + Q_2 + H_5 + H_3) - (L_3 + L_2 + G_5 + G_3)(2Q_3 + Q_1 + Q_2)}$$

$$v_{g_3} = \frac{2(J_3-J_2)-u_{g_3}(2L_3+L_1+L_2)}{2Q_3+Q_1+Q_2}$$

$$P_{g_3} = J_2 + \frac{u_{g_3}(L_3+L_2)+v_{g_3}(Q_3+Q_2)}{2}$$

$$\rho_{g_3} = \rho_{g_5} + \frac{(S_3+S_5)(P_{g_3}-P_{g_5})-(\bar{T}_3+\bar{T}_5)(z_3-z_5)}{2}$$

Step 2 through Step 7 are iterated until the calculated variables change by less than a prescribed amount, or until a maximum number of iterations is reached.

5.6.42 Subroutine SUMP1

Routine SUMP1 calculates the qualities i^t where

$$1^t = \sum_{j=1}^{K} A_{p_j} \rho_{p_j}$$

$$2^t = \sum_{j=1}^{K} A_{p_j} \rho_{p_j} u_{p_j}$$

$$3^t = \sum_{j=1}^{K} A_{p_j} \rho_{p_j} v_{p_j}$$

$$4^t = (\gamma-1) \sum_{j=1}^{K} A_{p_j} \rho_{p_j} \left[(u_g - u_{p_j})^2 + (v_g - v_{p_j})^2 - \frac{2}{3} \frac{C_{p_g}}{Pr} \frac{E_{p_j} u_{p_j}}{b_{p_j} \left(\frac{P_g}{\rho_g} \right)^N} \right]$$

5.6.43 Subroutine SUMP2

Subroutine SUMP2 calculates the same quantities as described above in SUMP1. Routine SUMP2 also calculates the additional quantities i^t where K is replaced by K-1, which are required by the k^{th} particle boundary point calculation.

5.6.44 Subroutine TAFN

This subroutine calculates an angle averaged tangent using the relation:

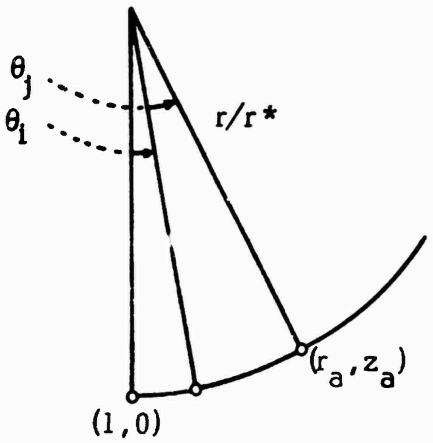
$$TA = \tan\left[\frac{1}{2}\left(\arctan X + \arctan Y\right)\right]$$

5.6.45 Subroutine WALL

This subroutine makes available five options for the specification of nozzle wall contours. Either a cone, parabola, arc, contour, or, a spline fit curve may be attached tangentially to a circular arc. The circular arc is specified as follows:

Symbol	Definition
θ_i	Angle defining the downstream end of the arc
θ_j	Angle defining the upstream end of the arc
r/r^*	Normalized arc radius

where



One hundred points are generated along the arc for equally incremented values of angle, θ:

$$r = 1 + r/r^* \ (1 - \cos \theta), \qquad \theta_i = \theta \leq \theta_j$$

$$x = r/r^* \sin \theta$$

Five options are specified below for attaching wall contours tangetially to this circular arc. The cone option requires the nozzle expansion ratio as input. The parabola and arc options require the exit point coordinates, r_e and x_e, as input. The spline fit option requires a smooth table of wall coordinates, r_i and x_j, and the exit angle at the nozzle lip, θ_e, as input.

Option 1, Cone

$$r_c = \sqrt{\epsilon}$$

$$x_c = (r_e - r_a)\text{ctn}\,\theta + x_a$$

Option 2, Parabola

Two hundred points are generated at equally incremented values of x such that:

$$r = a + \sqrt{b(x - c)}$$

where

$$a = \frac{r_e^2 - r_a^2 - 2r_a(x_e - x_a)\tan\theta_j}{2\left[r_e - r_a - (x_e - x_a)\tan\theta_j\right]}$$

$$b = 2(r_a - a)\tan\theta_j$$

$$c = x_e - \frac{(r_e - a)^2}{b}$$

Option 3, Arc

Two hundred points are generated at equally incrmented values of angle such that:

$$r = r_a + \bar{R}(\cos\theta - \cos\theta_j)$$

$$x = x_a + \bar{R}(\sin\theta_j - \sin\theta)$$

where

$$\bar{R} = \frac{(x_e - x_a)^2 + (x_e - r_a)^2}{2\left[(x_e - x_a)\sin\theta_j - (r_e - r_a)\cos\theta_j\right]}$$

and the increment on angle is

$$\Delta\theta = \frac{\theta_e - \theta_j}{200}$$

where

$$\theta_e = \sin^{-1}\left[\sin\theta_j - \frac{1}{R}(x_e - x_a)\right]$$

Option 4, Spline Fit

Two hundred points will be placed along a curve input in tabular form by a cubic spline fit method (see subroutines CUBIC and SLP). Derivatives are calculated from the input values for θ_j and the exit nozzle lip angle, θ_e. This curve must connect tangentially to the circular arc described above.

Option 5, Cone (r_e, x_e input)

$$\theta_j^{(0)} = 0$$

then for i = 1, . . . 5

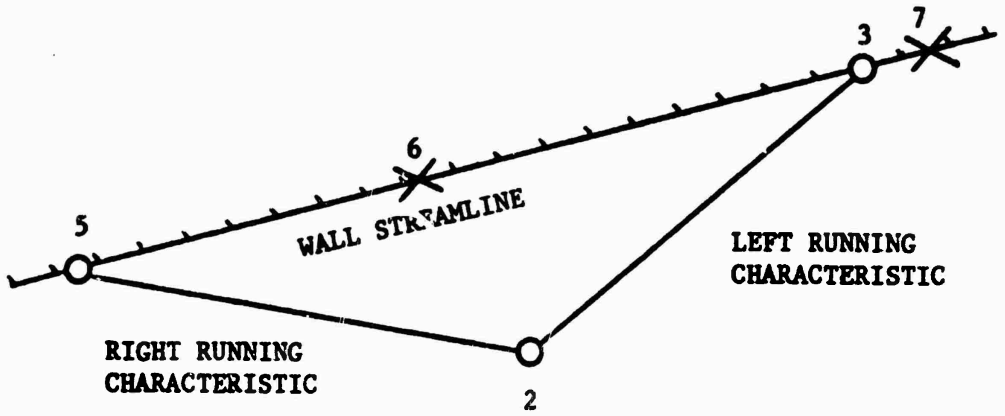
$$r_a^{(i)} = 1 + \frac{r}{r^*} (1 - \cos \theta_j^{(i-1)})$$

$$x_a^{(i)} = \frac{r}{r^*} \sin \theta_j^{(i-1)}$$

$$\theta_j^{(i)} = \arctan \left(\frac{r_e - r_a^{(i)}}{x_e - x_a^{(i)}} \right)$$

5.6.46 Subroutine WLPT

This subroutine uses two known points to calculate a wall point.



Properties at point 3 are computed from properties at points 2 and 5. The wall is necessarily a gas streamline. It is assumed that no particles are present at points 2 and 5.

STEP 1

Each variable at point 3 is set equal to the average of the corresponding variable at points 2 and 5.

STEP 2

Previous values of point 3 are saved, then

$$\bar{\kappa} = T\,A\,(\kappa_2, \kappa_3).$$

STEP 3

$$C_{g_3} = \frac{r_7 - r_6}{z_7 - z_6}$$

$$z_3 = \frac{r_6 - r_2 + \bar{\kappa}\, z_2 - C_{g_3}\, z_6}{\bar{\kappa} - C_{g_3}}$$

$$q_3 = \frac{z_3 - z_6}{z_7 - z_6}$$

If

$q_3 > 1 + \epsilon_w$ — select new points 6 and 7 upstream one point and restart STEP 3.

If

$q_3 < -\epsilon_w$ — select new points 6 and 7 downstream one point and restart STEP 3.

If

$-\epsilon_w \leq q_3 \leq 1 + \epsilon_w$ — proceed.

$$r_3 = r_6 + q_3(r_7 - r_6)$$

STEP 4

$$u_{g_3} = \frac{2(J_1 - J_2)}{L_3 + L_2 + G_5 + G_3 + C_{g_3}(Q_3 + Q_2 + H_5 + H_3)}$$

$$v_{g_3} = u_{g_3} C_{g_3}$$

$$P_{g_3} = J_1 - \frac{u_{g_3}(G_5 + G_3) + v_{g_3}(H_5 + H_3)}{2}$$

$$\rho_{g_3} = \rho_{g_5} + \frac{(S_3 + S_5)(P_{g_3} - P_{g_5})}{2}$$

Steps 2 through 4 are iterated until the calculated variables change by less than a prescribed amount, or until a maximum number of iterations is reached.

5.6.47 Subroutine XSLP

This subroutine is identical to subroutine SLP (Section 5.5.15).

5.7 Link 60 Subroutines

This overlay and subroutines contain the TBL Module. The subroutine write-ups are from Reference 5 with few exceptions.

Only a minimum of modifications have been made to the TBL module. These modifications deal mainly with the linkages between the various modules of the SPP code, ØDE and TD2P, and the TBL module.

5.7.1 Program Link 60

This subprogram is mainly intended to facilitate the conversion of the SPP code overlay structure to other computer systems. The only function of this routine is to call subroutine TBL.

5.7.2 Subroutine BARCON

Subroutine BARCON accepts initial conditions and executes a Runge-Kutta solution for the boundary layer along the contour. The initial conditions are set: $\phi = \phi_o$, $\theta = \theta_o$, at $x = x_o$. A step-size Δx, used to increment the axial coordinate x, is determined from input quantity Δx_{max} and the local entires in the x-table. Having two first order ordinary differential equations $\left| d\theta/dx = f(x, \theta, \phi) \text{ and } d\phi/dx = g(x, \theta, \phi) \right|$ and using subroutine BARPRO to evaluate the derivatives, a four term Runge-Kutta numerical solution is used.

1. Evaluate

$$f_1 = \frac{d\theta}{dx}\Big|_{x_o} \quad , \quad g_1 = \frac{d\phi}{dx}\Big|_{x_o}$$

Set

$$\theta = \theta_o + \frac{\Delta x}{2} (f_1) \quad , \quad \phi = \phi_o + \frac{\Delta x}{2} (g_1)$$

2. Evaluate

$$f_2 = \frac{d\theta}{dx}\Big|_{x_o + \frac{\Delta x}{2}} \quad , \quad g_2 = \frac{d\theta}{dx}\Big|_{x_o + \frac{\Delta x}{2}}$$

Set

$$\theta = \theta_o + \frac{\Delta x}{2} (f_2) \quad , \quad \phi = \phi_o + \frac{\Delta x}{2} (g_2)$$

3. Evaluate

$$f_3 = \frac{d\theta}{dx}\Big|_{x_o + \frac{\Delta x}{2}} \quad , \quad g_3 = \frac{d\phi}{dx}\Big|_{x_o + \frac{\Delta x}{2}}$$

Set

$$\theta = \theta_o + \Delta x (f_3) \quad , \quad \phi = \phi_o + \Delta x (g_3)$$

4. Evaluate

$$f_4 = \frac{d\theta}{dx}\Big|_{x_o + \Delta x} \quad , \quad g_4 = \frac{d\phi}{dx}\Big|_{x_o + \Delta x}$$

5. At $x + \Delta x$, θ and ϕ are then evaluated using

$$\theta = \theta_o + \frac{\Delta x}{6} (f_1 + 2f_2 + 2f_3 + f_4)$$

$$\phi = \phi_o + \frac{\Delta x}{6} (g_1 + 2g_2 + 2g_3 = g_4)$$

Now using these values of x, θ, ϕ as the initial values for the next interval, we repeat the above procedure on through to the end of the defined contour.

This subroutine also calculates and prints a table of normalized contour points corrected for displacement thickness. This print is made to assist in the preparation of a potential flow nozzle contour table for input ot the TD2P, TDK, or other method of characteristic programs. This dimensionless potential flow (adjusted) nozzle contour is calculated by the following relationships.

$$y_{t,POTENTIAL} = y_t \cdot y_{t,REAL} - \delta_t^*$$

$$y_{POTENTIAL} = [y \cdot y_{t,REAL} - \delta^* \cos \alpha]/y_{t,POTENTIAL}$$

$$x_{POTENTIAL} = [x \cdot y_{t,REAL} + \delta^* \sin \alpha]/y_{t,POTENTIAL}$$

where

$y_{t,REAL}$ = actual (geometrical) throat radius

x are XITAB

y are YITAB

and

α = arc tan dy/dx.

The nozzle throat is indicated by the subscript t.

5.7.3 Subroutine BARPRO

Subroutine BARPRO calculates the boundary layer, given energy thickness (ϕ) and momentum thickness (θ) and $d\phi/dx$ and $d\theta/dx$ along the contour at a point (x). The interpolation routine XNTERP is used to evaluate inviscid flow and contour properties and derivatives at point x. Subroutine ZETAIT is then entered, and the parameter ζ and boundary layer thickness Δ, δ, δ^* are computed. An iteration procedure is then used to calculate skin friction coefficient (C_f) and Stanton number (C_H).

1. An initial guess (C_{fa_g}) is made

2. Calculate $\overline{C}_f\overline{R}_\theta = \left(\frac{T_{aw}}{T_e}\right)^{1-m} C_{fa_g} R_\theta$

3. Subroutine CFEVAL is used to evaluate $\overline{C}_f$ as a function of $\overline{C}_f\overline{R}_\theta$

4. Calculate $\left(\frac{T_s}{T_{aw}}\right)$ and $C_{fa} = \frac{\overline{C}_f}{\left(\frac{T_{aw}}{T_e}\right)\left(\frac{T_s}{T_{aw}}\right)^m}$

5. A relative error comparison is made. If $\left|\frac{C_{fa}-C_{fa_g}}{C_{fa_g}}\right| < \text{TOLCFA}$ convergence is considered to be attained. If unconverged, a form of Wegstein's method is used to calculate a new guess (C_{fa_g}) and steps 2 through 5 are repeated up to a maximum of 50 iterations.

The derivatives $d\phi/dx$ and $d\theta/dx$ are now evaluated, from the differential equations, at the point x. For the point x at the end of a Runge-Kutta step (IND=1), total heat transfer and drag quantities are also computed.

This subroutine also prints the program output under control of the input variable, IPRINT.

This subroutine was also modified to print and write (punch) out the linkage data needed by the SPP master control module. In particular this routine calculates the force decrement due to a turbulent boundary layer using the equation

$$\Delta F = (2\pi y)^k \rho_e \frac{u_e^2}{2g} \cdot \theta \cdot \cos\alpha \left(1 - \frac{\delta^*}{\theta} \frac{P_e}{\frac{\rho_e u_e^2}{g}}\right)$$

where: $\alpha = \tan^{-1} \frac{dy}{dx}$

K = 0 for planar flow

K = 1 for axisymetric flow

and the decrement in I_{sp} is calculated from

$$\Delta I_{sp} = \Delta F/\dot{m}$$

where: $\dot{m}$ = mass flow from input or the TD2P module

ΔF = force decrement as calculated above

The above items are only printed out when the wall slope is positive.

5.7.4 Subroutine BARSET

This subroutine computes program constants and sets up C_p and H versus T tables and tables of P and T versus x for the inviscid flow. Based on the input table of C_p versus T, subroutine BMTAB is used to generate coefficients of a cubic spline fit. These coefficients are then used to compute piecewise partial integral values to be used by subroutine SEVAL which relates C_p, T, H, S and P. Subroutine GETPT is then used to evaluate pressure and temperature at each point of the Mach number table.

5.7.5 Subroutine BMTAB

This subroutine computes the coefficients of a piecewise cubic spline fit of tabular data.

5.7.6 Subroutine BMFIT

This subroutine computes the coefficients of a piecewise cubic spline fit of tabular data.

5.7.7 Subroutine CFEVAL

Subroutine CFEVAL evaluates the adiabatic skin friction coefficient, $\overline{C}_f$, as a function of $\overline{C}_f\overline{R}_\theta$, based on Coles' relationship. For values of $\overline{C}_f\overline{R}_\theta$, such that $0 < \overline{C}_f\overline{R}_\theta < 2.51$, $\overline{C}_f$ is evaluated from

$$\overline{C}_f = \frac{.009896}{(\overline{C}_f\overline{R}_\theta)^{.562}}$$

For values of $\overline{C}_f\overline{R}_\theta$, such that $2.51 < \overline{C}_f\overline{R}_\theta < 1982759.2$, $\overline{C}_f$ is evaluated from a piecewise cubic spline fit of log $(\overline{C}_f)$ vs log$(\overline{C}_f\overline{R}_\theta)$. All other values of $\overline{C}_f\overline{R}_\theta$ are considered out of range.

5.7.8 Subroutine DIRECT

Subroutine DIRECT controls the overall flow of the program. It calls subroutines which successively read in data, compute program constants, fit curves and then solve the boundary layer solution algorithm.

5.7.9 Subroutine EDITCP

This subroutine attempts to edit a table of C_p versus T from the boundary layer edge condition table of C_{p_e} versus T if the IFEDGE=1 or 2 option was selected in the $TBL namelist and if no C_p versus T table was input.

5.7.10 Subroutine FIIF

Subroutine FIFF defines the function (of "s") to be evaluated by the numerical integration routine INTZET. Three second degree coefficients (A_F, B_F, C_F) and an exponent value (m_f) are set by subroutine ZETAIT. Enthalpy value $\bar{h}$ is computed from

$$\bar{h} = A_F + B_F s + C_F s^2$$

Entering subroutine SEVAL with $\bar{h}$, we obtain $\bar{t}$. The appropriate form of the function is then evaluated:

$$\text{For } I_F = 1, \quad f(s) = \left(\frac{T}{\bar{t}}\right) s^{m_f} (1 - s)$$

$$I_F = 2, \quad f(s) = \left(\frac{T}{\bar{t}}\right) s^{m_f}$$

where $T/\bar{t}$ has been used for $\bar{\rho}/\rho$.

5.7.11 Subroutine GETPT

For a given Mach number, subroutine GETPT computes pressure P_e and temperature T_e from C_p, H and T relations. For constant specific heat ($\gamma = \gamma_o$)

$$T = \frac{T_o}{1 + \frac{\gamma - 1}{2} M^2}$$

$$P = \frac{P_o}{\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}}}$$

For variable C_p the following iteration is used

1. An initial guess is made for temperature T_g from

$$T_g = \frac{T_o}{1 + \frac{\gamma_o - 1}{2} M^2}$$

2. Using subroutine SEVAL, C_p and H are computed at temperature T_g.
3. The specific heat ratio γ is then evaluated from

$$\gamma = \frac{C_p}{C_p - \frac{R}{J}}$$

4. Temperature T_e is then calculated

$$T_e = \frac{2(H_o - H)}{\gamma R M^2}$$

5. A relative error comparison is then made. If

$$\left| \frac{T_e - T_g}{T_g} \right| < \text{TOLZME}$$

convergence is considered to be attained and subroutine SEVAL is used to evaluate pressure P_e at temperature T_e. If the convergence criterion is not satisfied, a form of Wegstein's method is used to calculate a new guess for T_g and steps 2 through 5 are repeated up to a maximum count of 50 iterations.

5.7.12 Subroutine INTZET

Subroutine INTZET, given a function definition and limits of integration (x_1 to x_2), uses a numerical technique based on the quintic spline to evaluate the integral of the function from x_1 to x_2. The interval over which the integration is to be performed is divided into 20 coarse subintervals. Since the limits are restricted to certain values (0, 1, ζ, $1/\zeta$), a check on the length of interval ($x_2 - x_1$) is made. If $(x_2 - x_1) < .10$, the function is evaluated at the 21 subinterval endpoints and the integral is approximated by the quintic technique, with a "one-sided Simpson's Rule" being used over the extreme subintervals.

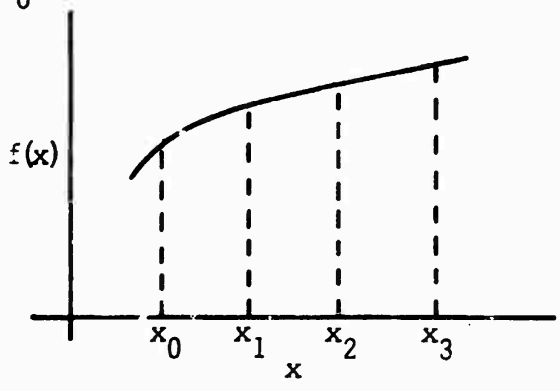
For $(x_2 - x_1) > .10$, the maximum value of the function within the interval is found, and the approximation is performed over the coarse steps up to a point where the value of the function is 20% of the maximum value. Beyond this point, the remaining coarse subintervals are each further divided into 3 fine subintervals. The function is evaluated at the endpoints of these fine subintervals. The integral approximation is completed using this finer step.

For a given function f(x) evaluated at equally spaced (Δx) points x_0, x_1, x_2, x_3, the quintic approximation for integration is given by:

$$\int_{x_1}^{x_2} f(x)\,dx = \frac{\Delta x}{24}\left(-f(x_0) + 13f(x_1) + 13f(x_2) - f(x_3)\right)$$

The "one-sided Simpson's Rule" is given by:

$$\int_{x_0}^{x_1} f(x)\,dx = \frac{\Delta x}{12}\left(5f(x_0) + 8f(x_1) - f(x_2)\right)$$



5.7.13 Subroutine QUITS

Subroutine QUITS is entered when an error has been detected in the input or if an unreasonable quantity has been detected during execution. It prints out appropriate error statements and the contents of labeled common storage.

5.7.14 Subroutine READSI

Subroutine READSI is a modified version of subroutine READIN of Reference 5. The name was changed to avoid a naming conflict with a subroutine by the same name in the ballistics module.

This routine reads the input data for the TBL module in the $TBL namelist as described in Volume III, Section 2.10. Subroutine READSI also checks the input data for consistancy and prints out dianogstics.

In addition to the above, this routine traps and/or reads any linkage data generated by the ØDE or TD2P modules.

5.7.15 Subroutine SEVAL

Subroutine SEVAL, using certain defined integral relationships, and a piecewise cubic curve fit of specific heat (C_p) as a function of temperature (T), for a given indicator value (Ind), evaluates the appropriate quantity: Pressure (P), entropy (S); enthalpy (H); temperature (T); specific heat (C_p).

For Ind = -1: P = P(T,S)
0: S = S(T,P)
1: H = H(T), $C_p = C_p(t)$
2: T = T(H), $C_p = C_p(t)$
3: T = T(S,P)

The basic relations are:

$$C_p = A + BT + CT^2 + DT^3$$

$$H(T) = H_i + \int_{T_i}^{T} C_p(t)\, dt$$

$$S(T) = S(T_o) + \int_{T_o}^{T} \frac{C_p(t)}{t}\, dt - R \ln\left(\frac{P}{P_o}\right)$$

5.7.16 Subroutine START

Subroutine START locates the sonic point (M=1) from the input Mach number and x tables and then computes initial values for momentum and energy thicknesses.

The table of Mach numbers is first scanned to locate an exact value of M=1. If none is found, the sonic point location is determined from the following iteration procedure.

1. Locate the i-th interval, in the table, which brackets M=1 and initially guess $x_g = x_i$.
2. Using subroutine XNTERP, evaluate M_g at x_g.
3. An absolute error comparison is made. If

$$| M_g - 1. | < \text{TOLZME}$$

convergence is considered to be attained and appropriate flow quantities are evaluated at x_g. If unconverged, a secant method is used to calculate a new guess for x_g and steps 2 and 3 are repeated up to a maximum of 50 iterations.

Using the flow quantities just computed, subroutine INTZET is used to evaluate integrals I_1 and I_2 for $\zeta = 1$. The shape factor δ^*/θ is then evaluated from equation (125 of Ref. 5). An iteration for the skin friction coefficient C_f is entered (similar to that in subroutine BARPRO), and a value of momentum thickness θ is computed. This value is assumed to be the initial value for θ and ϕ at x_g.

5.7.17 Subroutine TBL

Subroutine TBL initiates the execution of the TBL module by calling subroutine DIRECT.

5.7.18 Subroutine XNTERP

Subroutine XNTERP, given tables of n points of independent variable (x_i) and dependent variable (y_i), returns point value (y) and first derivative (y') for a given value of x.

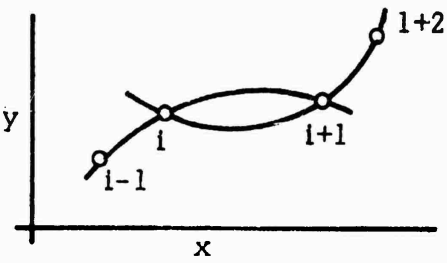
A local quintic spline interpolation is defined as follows: The independent variable table is searched until the points bracketing x are found ($x_i < x < x_{i+1}$). Using the four surrounding points (x_{i-1}, x_i, x_{i+1}, x_{i+2}), the coefficients of the parabola which passes through the three leftmost points and that which passes through the three rightmost points are obtained. For the given value x,

$$\bar{y}_L = c_1 + c_2 (x - x_{i-1}) + c_3 (x - x_{i-1})^2$$

$$\bar{y}_R = c_4 + c_5 (x - x_i) + c_6 (x - x_i)^2$$

$$\bar{y}'_L = c_2 + 2c_3 (x - x_{i-1})$$

$$\bar{y}'_R = c_5 + 2c_6 (x - x_i)$$



A cubic weighting function α and its derivative α' are defined as follows:

$$U(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

$$\alpha(x) = 3U^2 - 2U^3$$

$$\alpha'(x) = (6U - 6U^2)\, U'(x)$$

The dependent variable y and its derivative y' can now be evaluated.

5.7.19 Subroutine ZETAIT

Subroutine ZETAIT calculates the shape parameter ζ and boundary layer thicknesses Δ, δ and δ^* at a point x, for given values of θ and ϕ.

For $\zeta > 1$

$$\frac{\delta^*}{\theta} = \left(\frac{\zeta^{n/n} - I_2 - I_3}{I_1}\right)$$

$$\delta = \frac{\theta}{nI_1}$$

where

$$\zeta = \left[\frac{\phi}{\theta}\ \frac{I_1}{I_2' + I_3'/\zeta}\right]^{\frac{1}{n+1}}$$

For $\zeta < 1$

$$\frac{\delta^*}{\theta} = \left(\frac{1/n - I_6 - I_7}{I_4 + I_5}\right)$$

$$\delta = \frac{\theta}{n(I_4 + I_5)}$$

where

$$\zeta = \left[\frac{\phi}{\theta}\ \frac{I_4 + I_5}{I_1'}\right]^{\frac{1}{n+1}}$$

An iteration procedure is used to evaluate ζ.

1. An initial guess ζ_g is made
2. For the value of ζ_g (≥ 1 or < 1) appropriate functions of $\bar{\rho}/\rho$ have been defined, and the proper integrals are evaluated using subroutine INTZET
3. ζ is calculated by the appropriate formula above
4. A relative error comparison is made. If

$$\left|\frac{\zeta - \zeta_g}{\zeta_g}\right| < \text{TOLZET}$$

convergence is considered to be attained. If not converged, a form of Wegstein's method is used to calculate a new guess for ζ_g and steps 2 to 4 are repeated up to a maximum of 50 iterations.

The boundary layer thicknesses δ^* and Δ are then calculated.

6. REFERENCES

1. Gordon, S. and McBride, B. J., "Computer Program for Calculation of Complex Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks, and Chapman-Jouguet Detonations," NASA SP 273 (1971).

2. Barron, J. G., Jr., Cook, K. S., and Johnson, W. C., "Grain Design and Internal Ballistics Evaluation Program, Program No. 64101," Hercules Powder Co., Bachus Works, July 1967, AD 818321.

3. Nickerson, G. R., Coats, D. E., and Bartz, J. L., "The Two-Dimensional Kinetic (TDK) Reference Computer Program," Engineering and Programming Manual, Ultrasystems, Inc., Dec. 1973, prepared for Contract No. NAS9-12652, NASA JSC.

4. Nickerson, G. R., and Kliegel, J. R., "Axisymmetric Two-Phase Perfect Gas Performance Programs, Volume 1 Engineering and Programming Description" prepared for NASA MSC under Contract NAS9-4358, April 1967.

5. Weingold, H. D., "The ICRPG Turbulent Boundary Layer (TBL) Reference Program," prepared for ICRPG perf. Std. Working Group, July 1968.

6. Harry, D. P., Price, C. F., Small, K. R., and Taylor, D. E., "User's Manual For Nozzleless Rocket Motors Internal Ballistics Computer Programs," AFRPL TR-73-20, March 1973.

7. Svehla, R. H., "Estimated Viscosities and Thermal Conductivities of Gases at High Temperatures," NASA TR R-132, 1962.

8. Bird, R. B., Stweart, W. E., Lightfoot, E. N., "Transport Phenomena," John Wiley & Sons, Inc. New York 1960.

9. Levine, J. N., "Transpiration and Film Cooling Boundary Layer Computer Program, Vol. 1, Final Report," Contract NAS7-791, June 1971.

10. Kliegel, J. R., "Gas Particle Nozzle Flows," Ninth Symposium on Combustion, Academic Press, Inc. (New York and London), 1963, pp. 811-826.

11. Priem, R. J., Heidmann, M. F., "Propellant Vaporization as a Design Criterion for Rocket-Engine Combustion Chambers," NASA TR R-67, 1960.

Appendix A - Conversions Aides

The SPP code was developed under the rules and limitations of the SCOPE 3.3 operating systems of CDC 6000 computers. The following notes are intended to aid the conversion of the SPP code to other computers and/or operating systems.

1) Subroutine FIND of Section 5.1.6 is not the same routine as described in Section 5.6.5. On operating systems which poses a problem, it is suggested that the name of the routine of 5.6.5 be changed to FINE. This routine is referred to in subroutines ACØMP, ØNED, PARTIL, and TRACE.

2) Subroutine ERROR of Section 5.4.8 is not the same as the routine described in 5.6.37. On operating systems where this poses a problem it is suggested that the routine of 5.4.8 should be changed to the name (external reference) of ERRØS. This routine (5.4.8) is only referred to in subroutine READIN.

3) A possible conflict in labeled common areas occurs in LINK41 and LINK50 routines. Labeled common blocks by the name NAMEW occur in both of these overlays, however, these are not intended to be the same. It is suggested that the common block in subroutine ØDKINP be changed to the name NAMW to avoid this possible conflict.

4) Contrary to the rules of FØRTRAN IV, EQUIVALENCE statements can sometime increase the size of labeled common areas. This can cause problems on some operating systems. In particular the labeled common block FNALBT in subroutine DERIV can be made to be 4 locations (cells) longer than defined in subroutine MAIN1D. If this causes problems, the length of this common block should be extended by 4 dummy locations (which are not used) in subroutines MAIN1D, FLU, and IAUX.

Other minor corrections to the SPP code which will update the SPP code to the 1.2 version are:

a) Moving the labeled common block DØLØØPS into the ØDK routine.

b) Adding the variable CPSØL to the $TD2 namelist.

c) Adding the test at approximately line 65 of subroutine TD2P of
IF (FTD2P .EQ. 0.0) CPSØL = 0.0

d) Changing the test is subroutine AGP from
IF (FTD2P .EQ.0) GØ TØ 12
to: IF(CPSØL .EQ.0) GØ TØ 12

The conversion of the SPP code to the Univac 1108, EXEC 8 operation system took approximately one man day of effort. However, it is recognized by long and painful years of experience that the authors of a particular code have much less trouble converting codes than those unfamiliar with the particular program in question. Therefore, questions concerning the SPP code should be directed to the Air Force contracting officier in charge of the SPP code and responses will be made through him.

Other notes concerning conversions of the SPP code are as follows for Univac and IBM users.

I) The cards with ØVERLAY (SPP,i,j) should be removed.

II) The cards with PRØGRAM LINKij should be changed to SUBRØUTINE LINKij and a RETURN card should be added just before the END card.

III) The calls to ØVERLAY (3LSPP,i,j) should be changed to CALL LINKij.

It is hoped that these brief notes will substantially aid in the conversion of the SPP code to other computer and/or operating systems for which the SPP code was not designed for.